A Logical Method for Policy Enforcement over Evolving Audit Logs

Deepak Garg, Limin Jia, Anupam Datta

February 9, 2011

(revised February 24, 2011)

CMU-CyLab-11-002

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **24 FEB 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE
A Logical Method for Policy Enforcement over Evolving Audit Logs

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Carnegie Mellon University,CyLab,Pittsburgh,PA,15213

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We present an iterative algorithm for enforcing policies represented in a first-order logic which can, in particular, express all transmission-related clauses in the HIPAA Privacy Rule. The logic has three features that raise challenges for enforcement | uninterpreted predicates (used to model subjective concepts in privacy policies), real-time temporal properties, and quantification over infinite domains (such as the set of messages containing personal information). The algorithm operates over audit logs that are inherently incomplete and evolve over time. In each iteration, the algorithm provably checks as much of the policy as possible over the current log and outputs a residual policy that can only be checked when the log is extended with additional information. We prove correctness and termination properties of the algorithm. While these results are developed in a general form, accounting for many different sources of incompleteness in audit logs, we also prove that for the special case of logs that maintain a complete record of all relevant actions, the algorithm effectively enforces all safety and co-safety properties. The algorithm can significantly help automate enforcement of policies derived from the HIPAA Privacy Rule.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**52**	

A Logical Method for Policy Enforcement over Evolving Audit Logs[*]

Deepak Garg Limin Jia Anupam Datta

Technical Report CMU-CyLab-11-002

Revision of February 24, 2011

Abstract

We present an iterative algorithm for enforcing policies represented in a first-order logic, which can, in particular, express all transmission-related clauses in the HIPAA Privacy Rule. The logic has three features that raise challenges for enforcement — uninterpreted predicates (used to model subjective concepts in privacy policies), real-time temporal properties, and quantification over infinite domains (such as the set of messages containing personal information). The algorithm operates over audit logs that are inherently incomplete and evolve over time. In each iteration, the algorithm provably checks as much of the policy as possible over the current log and outputs a residual policy that can only be checked when the log is extended with additional information. We prove correctness and termination properties of the algorithm. While these results are developed in a general form, accounting for many different sources of incompleteness in audit logs, we also prove that for the special case of logs that maintain a complete record of all relevant actions, the algorithm effectively enforces all safety and co-safety properties. The algorithm can significantly help automate enforcement of policies derived from the HIPAA Privacy Rule.

1 Introduction

Organizations, such as hospitals, banks, and universities, that collect, use, and share personal information have to ensure that they do so in a manner that respects the privacy of the information subjects. In fact, designing effective processes to audit transmission and access logs to ensure compliance with privacy regulations, such as the Health Insurance Portability and Accountability Act (HIPAA) [32], has become one of the greatest challenges facing organizations today (see, for example, a recent survey from Deloitte and the Ponemon Institute [15]). State-of-the-art commercial tools such as the FairWarning [1] allow auditors to mine access and transmission logs and flag potential violations of policy, but do not help decide which flagged items are actual violations, even though privacy legislation often lays down objective criteria to make such decisions. We address

[*]This work was partially supported by the U.S. Army Research Office contract "Perpetually Available and Secure Information Systems" (DAAD19-02-1-0389) to Carnegie Mellon CyLab, the NSF Science and Technology Center TRUST, the NSF CyberTrust grant Privacy, Compliance and Information Risk in Complex Organizational Processes, the AFOSR MURI Collaborative Policies and Assured Information Sharing, and HHS Grant no. HHS 90TR0003/01. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

this challenge by developing a novel, logic-based method for computer-assisted enforcement of policies. This method can be used to enforce a rich class of privacy and security policies that include, in particular, real privacy regulations like HIPAA.

Policy Specification The first challenge for policy enforcement is formal specification of real policies. This challenge was addressed in our prior work on PrivacyLFP [16], an expressive first-order temporal logic, in which we represented formally all transmission-related clauses of the HIPAA and GLBA Privacy Laws. PrivacyLFP is more expressive than prior logics considered for expressing policies, including propositional temporal logics [8, 18] and first-order metric temporal logic [10].

Building on the prior work on specification of privacy laws in PrivacyLFP, this paper presents an algorithm for enforcing policies represented in the logic, through iterative analysis of audit logs, which we assume are collected independently and provided to us. The policy enforcement algorithm and the formulation and proof of its properties are the main contribution of this paper.

Three concepts in privacy legislation (and PrivacyLFP) make mechanical enforcement particularly difficult; we discuss these concepts briefly. First, PrivacyLFP includes *uninterpreted or subjective predicates* to model subjective parts of privacy laws. For example, HIPAA allows transmission of protected health information about an individual from a hospital to a law enforcement agency if the hospital believes that the death of the individual was suspicious. Such beliefs are represented using uninterpreted predicates because the truth value of these predicates cannot, in general, be determined mechanically.

Second, PrivacyLFP allows first-order quantification over infinite domains (e.g., the set of messages or the set of time points). For example, many HIPAA clauses are of the form $\forall p_1, p_2, m.(\mathtt{send}(p_1, p_2, m) \supset \phi)$ where p_1 and p_2 are principals and m is a message. Note that this formula quantifies over the infinite set of messages, so if an enforcement algorithm were to blindly instantiate the quantifiers with all possible values in the domain, then it will not terminate. However, only a finite number of messages are relevant in determining the truth value of this formula. This is because the number of messages transmitted from a hospital is finite and hence the predicate $\mathtt{send}(p_1, p_2, m)$ is true for only a finite number of substitutions for the variable m (and similarly for p_1 and p_2). To ensure that the number of relevant substitutions for every quantified variable is finite, we use the idea of *mode checking* from logic programming [4], and restrict the syntax of quantifiers in PrivacyLFP slightly. The finite substitution property for quantified variables over infinite domains is defined in Section 4, and ensures termination of our policy enforcement algorithm. The restriction on quantification does not significantly limit representation of HIPAA clauses, a claim we justify in Section 6.

Third, the representation of one transmission-related clause – Section 6802(c) – of the GLBA Privacy Law forces PrivacyLFP to include fixpoint operators. In this paper, we do not consider fixpoints because the representation of most privacy legislation including all of HIPAA does not require fixpoints. We note that including the least fixpoint operator in our algorithm may not be difficult, but supporting the greatest fixpoint may require a substantial effort.

Audit logs Another significant challenge in mechanical enforcement of privacy policies is that the logs maintained by organizations may be incomplete, i.e., they may not contain enough information to decide whether or not the policy has been violated. For instance, in the absence of human input, a machine may not be able to decide whether any instance of a predicate that refers to subjective beliefs is true or not. Similarly, we may not be able to predict whether a predicate holds in the future or not. As an important contribution, we observe that such possibly incomplete logs can be abstractly represented as three-valued, *partial structures* that map each atomic formula to either true, false, or unknown [13, 19]. We define the semantics of our logic over such structures. Further, by designing our enforcement algorithm to work with partial structures in general, we provide a uniform account of policy enforcement with different forms of log incompleteness.

We explicitly discuss in Section 5.2 a special case of partial structures that are complete up to a point of time. This instance corresponds to the standard model of traces used in prior work on enforcement of temporal privacy properties [10]. We show that on such structures, our algorithm yields a method to find violations of safety properties [2] and satisfactions of co-safety properties [11] at the earliest possible time, as may be expected.

A second important observation is that, in practice, structures evolve over time by gathering more information. We formalize this growth as a natural order, $\mathcal{L}_1 \geq \mathcal{L}_2$ (structure $\mathcal{L}_1$ *extends* structure $\mathcal{L}_2$), meaning that $\mathcal{L}_1$ has more information than $\mathcal{L}_2$. We present a general definition of extension of partial structures, which encompasses, in particular, notions of temporal (actions are added to the end of a trace) and spatial (distributed logs are merged) extensions.

Policy Enforcement As our central contribution, we propose an iterative process for privacy policy enforcement. At each iteration, our algorithm takes as inputs a structure $\mathcal{L}$ abstracting the then-current audit log and a policy specification φ, verifies parts of the policy that depend solely on the given structure, and outputs a residual policy φ' that contains all the conditions that need to be verified when more information becomes available. We write $\mathsf{reduce}(\mathcal{L}, \varphi) = \varphi'$ to denote one iteration of our reduction algorithm. The residual policy φ' is checked on extensions of $\mathcal{L}$.

Our reduction algorithm has several desirable properties that we prove formally. First, the algorithm always *terminates*. As noted earlier, the finite substitution property for variables quantified over infinite domains is crucial for termination. Second, it is *correct*: given a structure $\mathcal{L}$ and a policy φ, any extension of $\mathcal{L}$ satisfies the policy φ if and only if it satisfies the residual formula φ'. Third, it is *minimal*: the residual formula only contains atoms whose truth value cannot be determined from the structure.

Our algorithm has been designed for after-the-fact (a-posteriori) audit, not runtime verification. However, as shown in Section 5.2, for the specific case of policies that do not contain any subjective predicates or future obligations, the algorithm may be executed at each privacy-relevant event to act as a runtime monitor, if all relevant past system logs can be provided to it.

Application to HIPAA Our technical results have important implications for enforcing practical privacy policies, in particular, the HIPAA Privacy Rule. As discussed in Section 6, not only can our algorithm be used to automatically instantiate all quantifiers in all 84 transmission-related clauses of HIPAA, but it can also automatically discharge the large percentage of non-subjective atoms in instantiated clauses. For example, we estimate that in 17 of the 84 clauses, all atoms can be discharged automatically, and in 24 other clauses, at least 80% of the atoms can be discharged automatically.

Summary of Contributions In summary, the contributions of this paper are:

- An iterative algorithm for enforcing policies represented in PrivacyLFP, a rich logic with quantification over infinite domains, and formulation and proofs of the algorithm's properties (Section 4)

Objective predicates	p_O		
Subjective predicates	p_S		
Objective atoms	P_O	$::=$	$p_O(t_1, \ldots, t_n)$
Subjective atoms	P_S	$::=$	$p_S(t_1, \ldots, t_n)$
Formulas	α, β	$::=$	$P_O \mid P_S \mid \top \mid \bot \mid$ $\alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2 \mid \neg\alpha \mid$ $\forall\vec{x}.(c \supset \alpha) \mid \exists\vec{x}.(c \wedge \alpha) \mid$ $\downarrow x.\alpha \mid \alpha \,\mathrm{S}\, \beta \mid \alpha \,\mathrm{U}\, \beta \mid$ $\boxminus\alpha \mid \Box\alpha$
Restrictions	c	$::=$	$P_O \mid \top \mid \bot \mid c_1 \wedge c_2 \mid$ $c_1 \vee c_2 \mid \exists x.c$

Figure 1: Timed First-order Temporal Logic with Restricted Quantifiers

- Use of mode analysis from logic programming to ensure that infinite quantifiers result only in a finite number of relevant substitutions (Section 4)
- A formal model of incomplete audit logs as three-valued structures (Section 3)

Organization In Section 2, we review PrivacyLFP to the extent needed for this paper. Section 3 presents partial structures and defines the semantics of PrivacyLFP over them. Section 4 presents our policy enforcement algorithm and its properties. Section 5 discusses the behavior of our algorithm on structures that are complete and those that are complete up to a point of time. In the latter case, we also present associated results about enforcement of safety and co-safety properties. Section 6 describes how the work in this paper applies to the HIPAA Privacy Rule. Section 7 provides a detailed comparison with related work and Section 8 presents conclusions and directions for future work.

2 Policy Logic

We use PrivacyLFP [16] to represent policies, but restrict the syntax of first-order quantifiers slightly to facilitate enforcement and drop fixpoint operators. PrivacyLFP consists of an outer policy logic with connectives of temporal logic and an inner, equally expressive sublogic without connectives of temporal logic to which the outer syntax is translated. Our enforcement algorithm works only with the inner sublogic. In this section we review both the outer syntax and the sublogic, as well as the translation.

2.1 Syntax of the Policy Logic

The syntax of our policy logic is shown in Figure 1. We distinguish two classes of predicate symbols: 1) *objective predicates*, denoted p_O, that can be decided automatically using information from logs or using constraint solvers and 2) *subjective predicates*, denoted p_S, that require human input to resolve. Both classes of predicates are illustrated in examples later. An atom is a predicate applied to a list of terms (terms are denoted t). Based on the class of its predicate, an atom is also classified as either objective or subjective, written P_O and P_S, respectively.

Propositional connectives $\top$ (true), $\bot$ (false), $\wedge$ (conjunction), $\vee$ (disjunction), and $\neg$ (negation) have their usual meanings. Anticipating the requirements of the enforcement algorithm of Section 4, first-order quantifiers $\forall\vec{x}.(c \supset \alpha)$ and $\exists\vec{x}.(c \wedge \alpha)$ in the logic are forced to include a formula c called a *restriction*. By definition, $\forall x.(c \supset \alpha)$ is true iff all instances of $\vec{x}$ that satisfy c, also satisfy α. ($\exists\vec{x}.(c \wedge \alpha)$ has a similar definition.) To make enforcement tractable, we require that the set of instances of $\vec{x}$ satisfying c be computable. This is ensured by limiting c to a reduced class of formulas that, in particular, excludes subjective predicates (see the syntax of c in Figure 1), and through a static analysis that we describe in Section 4.

Further, our logic includes standard connectives of linear temporal logic (LTL) [23] that provide quantification over the sequence of states in a system, relative to a current state: $\alpha\,\mathrm{S}\,\beta$ (β holds at some state in the past and α holds since then), $\alpha\,\mathrm{U}\,\beta$ (β holds at some state in the future and α holds until then), $\boxminus\alpha$ (α holds at all states in the past) and $\Box\alpha$ (α holds at all states in the future). Other temporal operators can be defined, e.g., $\diamondminus\alpha = \top\,\mathrm{S}\,\alpha$ (α holds at some state in the past) and $\Diamond\alpha = \top\,\mathrm{U}\,\alpha$ (α holds at some state in the future).

Finally, to represent clock time, which often occurs in privacy policies, we assume that each state of a system has a time point associated with it. Time points, denoted τ, are elements of $\mathrm{T} = \{x \in \mathrm{R} \mid x \geq 0\} \cup \{\infty\}$. They measure clock time elapsed from a fixed reference point and order states linearly. Relations between time points are captured in logical formulas using the *freeze* quantifier $\downarrow x.\alpha$ of timed propositional temporal logic (TPTL) [3], which means "α holds with the current time bound to x." (Examples below illustrate the quantifier.) Since we have no occasion to reason explicitly about states, we identify a state with the time point associated with it, and use the letter τ and any of the terms "state", "time point", "time", and "point" to refer to both states and time points. We make the assumption that on any trace there are only finitely many time points between two given finite time points.

We illustrate the syntax of our logic through two examples that are based on the formalization of HIPAA in PrivacyLFP. These examples are also used later in the paper.

Example 2.1. As a first example, we represent in our logic the following policy about disclosure (transmission) of health information from one entity (e.g., a hospital or doctor) to another.

> An entity may send an individual's protected health information (phi) to another entity only if the receiving entity is the patient's doctor and the purpose of the transmission is treatment, or the individual has previously consented to the transmission.

Our formalization assumes that each transmitted message m is *tagged* by the sender (in a machine-readable format) with the names of individuals whose information it carries as well the attributes of information it carries (attributes include "address", "social security number", "medications", "medical history", etc.). The predicate $\texttt{tagged}(m, q, t)$ means that message m is tagged as carrying individual q's attribute t. Tagging may or may not reflect accurately the content of the message. Similarly, we assume that each message m is labeled in a machine readable format with a purpose u (e.g., "treatment", "healthcare", etc.). This is represented by the predicate $\texttt{purp}(m, u)$. Because we assume that name and attribute tags as well as purpose labels are machine readable, both `tagged` and `purp` are *objective predicates* – their truth or falsity can be checked using a program.

Attributes are assumed to have a hierarchy, e.g., the attribute "medications" is contained in "medical history". This is formalized as the predicate $\texttt{attr\_in}(\text{medications}, \text{medical-history})$.

We assume that the hierarchy can be mechanically checked, so $\mathtt{attr\_in}$ is an objective predicate. The predicate $\mathtt{purp\_in}(u, u')$ means that purpose u is a special case of purpose u', e.g., $\mathtt{purp\_in}(\mathrm{surgery}, \mathrm{treatment})$. In contrast to attributes, we assume that the purpose hierarchy cannot be computed, so $\mathtt{purp\_in}$ is a subjective predicate. In an enforcement system, it must be checked through human input.

Finally, each action or fact that can be recorded in a system log (such as sending a message or that Alice is in role doctor) is represented as an objective predicate. For this example we need three objective predicates: $\mathtt{send}(p_1, p_2, m)$ meaning that entity p_1 sends message m to entity p_2, $\mathtt{consents}(q, a)$ which means that individual q consents to the action a, and $\mathtt{inrole}(p, r)$ which means that principal p is in role r. Here, the only action consented to is $\mathtt{sendaction}(p_1, p_2, (q, t))$, which corresponds to p_1 sending to p_2 a message containing information about q's attribute t.

The above policy can be formalized in our logic as follows.

$$
\begin{array}{l}
\alpha_{pol1} = \\
\quad \forall p_1{,}p_2, m, u, q, t.\ (\mathtt{send}(p_1, p_2, m) \wedge \mathtt{purp}(m, u) \wedge \\
\qquad\qquad\qquad\qquad \mathtt{tagged}(m, q, t) \wedge \mathtt{attr\_in}(t, \mathit{phi})) \\
\qquad \supset (\mathtt{inrole}(p_2, \mathtt{doc}(q)) \wedge \mathtt{purp\_in}(u, \mathit{treatment})) \\
\qquad\quad \vee \diamondminus \mathtt{consents}(q, \mathtt{sendaction}(p_1, p_2, (q, t)))
\end{array}
$$

In words, if entity p_1 sends to entity p_2 a message m, m is tagged as carrying attribute t of individual q, where t is a form of phi (protected health information), and m is labeled with purpose u, then either p_2 (the recipient) is a doctor of q (atom $\mathtt{inrole}(p_2, \mathtt{doc}(q))$) and u is a type of treatment, or q has consented to this transmission in the past (last line of α_{pol1}). The temporal operator $\diamondminus$ is used to indicate that the consent may have been given by q in some earlier state. Also, the universal quantifier in the formula above carries a restriction ($\mathtt{send}(p_1, p_2, m) \wedge \mathtt{purp}(m, u) \wedge \mathtt{tagged}(m, q, t) \wedge \mathtt{attr\_in}(t, \mathit{phi})$), as required by our syntax. The technical reason for including restrictions is explained in Section 4.

Example 2.2. Our next example is a policy governing entity response to an individual's request for her own information.

> If an individual requests her information from an entity, then some administrator in the records department of the entity must respond to the individual at the earliest feasible time, but not later than 30 days after the request.

To represent this policy we need one more objective predicate, $\mathtt{req}(p, t)$, which means that individual p requests information about attribute t from her record. Further, we need two new subjective predicates: $\mathtt{contains}(m, q, t)$ (message m contains attribute t of individual q) and $\mathtt{ftr}(p, t)$ (it is feasible to respond to individual p with attribute t at the current time). The latter clearly requires human input to resolve, because "feasibility" cannot be defined mechanically, while the former requires human input because we assume that message payloads may contain natural language text.

The logical specification of this policy is shown below:

$$
\begin{array}{l}
\alpha_{pol2} = \\
\quad {\downarrow}\tau.\forall p{,}t.\ \mathtt{req}(p, t) \\
\qquad\quad \supset \neg\mathtt{ftr}(p, t) \\
\qquad\qquad \mathrm{U}\ {\downarrow}\tau'.\ \mathtt{in}(\tau', \tau, \tau + 30)
\end{array}
$$

$$\begin{aligned}\wedge\ \exists q, m.\ (&\texttt{inrole}(q, \text{records}) \wedge \texttt{send}(q, p, m) \wedge \\ &\texttt{contains}(m, p, t))\end{aligned}$$

The top-most quantifier $\downarrow\tau$ binds τ to the time at which a request occurs and, similarly, $\downarrow\tau'$ binds τ' to the time at which a response is sent. $\texttt{in}(\tau', \tau, \tau + 30)$, formally explained in Section 2.2, implies that $\tau' \leq \tau + 30$, thus enforcing the constraint that the response be sent within 30 days of the request, as required by the policy. The until operator $\mathcal{U}$ is used to include the obligation that it be infeasible to respond until the response is actually sent.

2.2 Translation to a Smaller Syntax

Policies expressed in PrivacyLFP's outer syntax can be translated into a smaller sublogic without temporal connectives and negation. This smaller syntax of formulas φ, ψ of the sublogic is shown below. Other syntactic categories such as restrictions c are not changed.

$$\begin{array}{lcl}\text{Formulas} \quad \varphi & ::= & P_O \mid P_S \mid \top \mid \bot \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \\ & & \forall\vec{x}.(c \supset \varphi) \mid \exists\vec{x}.(c \wedge \varphi)\end{array}$$

We surmount the absence of negation in the sublogic by *defining* for each formula φ a dual $\overline{\varphi}$ that behaves exactly as $\neg\varphi$ would. For defining duals of atoms, we assume that each predicate p has a dual $\overline{p}$ such that $p(t_1, \ldots, t_n)$ is true iff $\overline{p}(t_1, \ldots, t_n)$ is false (the relation between p and $\overline{p}$ is formalized in Section 3). We define $\overline{\varphi}$ by induction on φ, as in the representative clauses below (for the remaining clauses see Appendix A).

$$\begin{array}{ccc}\overline{p_O(t_1, \ldots, t_n)} & = & \overline{p_O}(t_1, \ldots, t_n) \\ \overline{\varphi \wedge \psi} & = & \overline{\varphi} \vee \overline{\psi} \\ \overline{\forall\vec{x}.(c \supset \varphi)} & = & \exists\vec{x}.(c \wedge \overline{\varphi}) \\ \overline{\exists\vec{x}.(c \wedge \varphi)} & = & \forall\vec{x}.(c \supset \overline{\varphi})\end{array}$$

Temporal connectives are translated to the sublogic by making time points (states) and the ordering relation between them explicit in first-order formulas in a standard way (see [16]). Briefly, we assume that for every predicate symbol in the logic there is a predicate of the same name in the sublogic, but with one extra argument of type time: $p(t_1, \ldots, t_n, \tau)$ in the sublogic means that $p(t_1, \ldots, t_n)$ holds at time τ in the logic. Further, assume that the new objective predicate $\texttt{in}(\tau, \tau_1, \tau_2)$ means that τ is an observed time point (in the trace of interpretation) satisfying $\tau_1 \leq \tau \leq \tau_2$. Finally, let $\Xi[\vec{t}/\vec{x}]$ denote the result of substituting the terms $\vec{t}$ for variables $\vec{x}$ in the syntactic entity Ξ. Then, representative clauses of the translation $(\bullet)^\tau$ of restrictions and formulas of the logic to those of the sublogic, indexed by a "current time" τ, are shown below (the full translation is shown in Appendix A):

$$
\begin{array}{rcl}
(p_O(t_1,\ldots,t_n))^\tau & = & p_O(t_1,\ldots,t_n,\tau)\\
(p_S(t_1,\ldots,t_n))^\tau & = & p_S(t_1,\ldots,t_n,\tau)\\
(\neg\alpha)^\tau & = & \overline{(\alpha)}^\tau\\
(\forall\vec{x}.(c\supset\alpha))^\tau & = & \forall\vec{x}.((c)^\tau\supset(\alpha)^\tau)\\
({\downarrow}x.\alpha)^\tau & = & (\alpha[\tau/x])^\tau\\
(\alpha\,\mathrm{S}\,\beta)^\tau & = & \exists\tau'.(\mathtt{in}(\tau',0,\tau)\wedge(\beta)^{\tau'}\\
 & & \quad\wedge(\forall\tau''.((\mathtt{in}(\tau'',\tau',\tau)\wedge\tau'\neq\tau'')\\
 & & \qquad\qquad\supset(\alpha)^{\tau''})))\\
(\alpha\,\mathrm{U}\,\beta)^\tau & = & \exists\tau'.(\mathtt{in}(\tau',\tau,\infty)\wedge(\beta)^{\tau'}\\
 & & \quad\wedge(\forall\tau''.((\mathtt{in}(\tau'',\tau,\tau')\wedge\tau''\neq\tau')\\
 & & \qquad\qquad\supset(\alpha)^{\tau''})))
\end{array}
$$

We briefly explain some of the clauses of the translation. In $({\downarrow}x.\alpha)^\tau$, x binds to the current time, which is τ; therefore, τ substitutes x in α in the translation. $\alpha\,\mathrm{S}\,\beta$ means that β is true at some time point in the past, which is captured by the existentially quantified variable τ' in the translation, and the restriction that $\mathtt{in}(\tau',0,\tau)$. Further, α should be true at all time points between τ' and now (τ); this is encoded as $\forall\tau''.((\mathtt{in}(\tau'',\tau',\tau)\wedge\tau''\neq\tau')\supset(\alpha)^{\tau''})$.

Example 2.3. In Section 2.1 we presented two sample policies, α_{pol1} and α_{pol2}. In general, we may wish to enforce each of these policies in each state. To express the phrase "in each state", we define an abbreviation: $\mathbf{G}\,\alpha=\forall\tau.(\mathtt{in}(\tau,0,\infty)\supset(\alpha)^\tau)$, which means that α holds at each time point τ. Then, using the translation above and simplifying slightly, we get:

$$
\begin{array}{l}
\mathbf{G}\,\alpha_{pol1}=\\
\forall\tau,p_1,p_2,m,u,q,t.\\
\quad(\mathtt{in}(\tau,0,\infty)\wedge\mathtt{send}(p_1,p_2,m,\tau)\wedge\mathtt{purp}(m,u,\tau)\wedge\\
\quad\ \mathtt{tagged}(m,q,t,\tau)\wedge\mathtt{attr\_in}(t,\mathit{phi},\tau))\\
\quad\ \supset((\mathtt{inrole}(p_2,\mathtt{doc}(q),\tau)\wedge\\
\qquad\quad\mathtt{purp\_in}(u,\mathit{treatment},\tau))\vee\\
\qquad(\exists\tau'.\ (\mathtt{in}(\tau',0,\tau)\wedge\\
\qquad\qquad\mathtt{consents}(q,\mathtt{sendaction}(p_1,p_2,(q,t)),\tau'))))
\end{array}
$$

$$
\begin{array}{l}
\mathbf{G}\,\alpha_{pol2}=\\
\forall\tau,p,t.\ (\mathtt{in}(\tau,0,\infty)\wedge\mathtt{req}(p,t,\tau))\\
\quad\supset\exists\tau',q,m.\\
\qquad((\mathtt{in}(\tau',\tau,\tau+30)\wedge\mathtt{inrole}(q,\mathrm{records},\tau')\wedge\\
\qquad\ \ \mathtt{send}(q,p,m,\tau'))\wedge\mathtt{contains}(m,p,t,\tau')\wedge\\
\qquad\forall\tau''.\ (\mathtt{in}(\tau'',\tau,\tau')\wedge\tau''\neq\tau')\\
\qquad\quad\supset\overline{\mathtt{ftr}}(p,t,\tau''))
\end{array}
$$

Note that all atoms, except those like $\mathtt{in}(\ldots)$ and $\tau''\neq\tau'$ that are introduced by the translation itself, have a new last argument, which is a time point. For certain predicates like `tagged`, `attr_in` and `purp_in`, whose truth is independent of time, this last argument is redundant. For instance, if $\mathtt{attr\_in}(t,t',\tau)$ for some τ, then $\mathtt{attr\_in}(t,t',\tau')$ for all τ'.

3 Partial Structures and Semantics

Next, we define *partial structures*, an abstraction of audit logs over which our enforcement algorithm (Section 4) works. We call our structures partial because they do not necessarily stipulate the truth or falsity of every atom, thus accurately reflecting the fact that audit logs may be incomplete in practice. We also illustrate, by virtue of example, various kinds of audit log incompleteness that our partial structures generalize. Finally, we define the *semantics* (meanings) of formulas of the sublogic on partial structures. This definition is used in Section 4 to state the correctness of our enforcement mechanism. Partial structures have been used, both explicitly and implicitly, in prior work on policy enforcement; we compare to such work in Section 7.

A *partial structure* (abbrev. structure) is a pair $\mathcal{L} = (\mathrm{D}_{\mathcal{L}}, \rho_{\mathcal{L}})$, where $\mathrm{D}_{\mathcal{L}}$, the domain, is a set of terms containing at least all possible time points T, and $\rho_{\mathcal{L}}$ is a total function from ground (variable-free) atoms of the logic to the *three-value* set $\{\mathtt{tt}, \mathtt{ff}, \mathtt{uu}\}$. We say that the atom P is true, false, or unknown in the structure $\mathcal{L}$ if $\rho_{\mathcal{L}}(P)$ is $\mathtt{tt}$, $\mathtt{ff}$, or $\mathtt{uu}$, respectively. In practice, the structure $\mathcal{L}$ may be defined using system logs (hence the notation $\mathcal{L}$), whence, $\mathrm{D}_{\mathcal{L}}$ would be the set of all terms (roles, principals, messages, attributes, time points, etc.) occurring in the logs and for every subjective atom P_S, $\rho_{\mathcal{L}}(P_S)$ would be $\mathtt{uu}$.

The semantics of our sublogic lift the definition of *truth* to formulas φ by induction on φ: we write $\mathcal{L} \models \varphi$ to mean that "φ is true in the structure $\mathcal{L}$". Restrictions c are a subsyntax of formulas φ, so we do not define the relation separately for them.

- $\mathcal{L} \models P$ iff $\rho_{\mathcal{L}}(P) = \mathtt{tt}$
- $\mathcal{L} \models \top$
- $\mathcal{L} \models \varphi \wedge \psi$ iff $\mathcal{L} \models \varphi$ and $\mathcal{L} \models \psi$
- $\mathcal{L} \models \varphi \vee \psi$ iff $\mathcal{L} \models \varphi$ or $\mathcal{L} \models \psi$
- $\mathcal{L} \models \forall \vec{x}.(c \supset \varphi)$ iff for all $\vec{t} \in \mathrm{D}_{\mathcal{L}}$ either $\mathcal{L} \models \overline{c}[\vec{t}/\vec{x}]$ or $\mathcal{L} \models \varphi[\vec{t}/\vec{x}]$
- $\mathcal{L} \models \exists \vec{x}.(c \wedge \varphi)$ iff there exists $\vec{t} \in \mathrm{D}_{\mathcal{L}}$ such that $\mathcal{L} \models c[\vec{t}/\vec{x}]$ and $\mathcal{L} \models \varphi[\vec{t}/\vec{x}]$

For dual atoms, we define $\rho_{\mathcal{L}}(\overline{P}) = \overline{\rho_{\mathcal{L}}(P)}$, where $\overline{\mathtt{tt}} = \mathtt{ff}$, $\overline{\mathtt{ff}} = \mathtt{tt}$, and $\overline{\mathtt{uu}} = \mathtt{uu}$. We say that a formula φ is *false* on the structure $\mathcal{L}$ if $\mathcal{L} \models \overline{\varphi}$. The following two properties hold:

1. Consistency: A formula φ cannot be simultaneously true and false in the structure $\mathcal{L}$, i.e., either $\mathcal{L} \not\models \varphi$ or $\mathcal{L} \not\models \overline{\varphi}$
2. Incompleteness: A formula φ may be neither true nor false in a structure $\mathcal{L}$, i.e., $\mathcal{L} \not\models \varphi$ and $\mathcal{L} \not\models \overline{\varphi}$ may both hold.

The first property follows by induction on φ. The second property follows from a simple example. Consider a structure $\mathcal{L}$ and an atom P such that $\rho_{\mathcal{L}}(P) = \mathtt{uu}$. Then, $\mathcal{L} \not\models P$ and $\mathcal{L} \not\models \overline{P}$.

Incompleteness in Practice We list below several ways in which system logs may be incomplete, and describe how each can be modeled in partial structures by varying the definition of $\rho_{\mathcal{L}}$.

- Subjective incompleteness: An audit log may not contain information about subjective predicates. This may be modeled by requiring that $\rho_{\mathcal{L}}(P_S) = \mathtt{uu}$ for every subjective atom P_S. We revisit subjective incompleteness in the context of our enforcement algorithm in Section 5.1.

- Future incompleteness: An audit log may not contain information about the future, which is necessary to enforce policies like that in Example 2.2. This may be modeled by assuming that for each time τ greater than the last point observed in $\mathcal{L}$, and for all p, $t_1, \ldots, t_n$, $\rho_{\mathcal{L}}(p(t_1, \ldots, t_n, \tau)) = \mathtt{uu}$. (Recall that in our translation of the outer logic, the last argument τ is the time at which the predicate's truth is tested.) We revisit future incompleteness in the context of our enforcement algorithm in Section 5.2.

- Spatial incompleteness: An audit log may not record all predicates. For instance, with reference to Example 2.1, it is conceivable that the predicates `send` and `inrole` are stored on separate sites. If we audit at the first site, information about `inrole` may be unavailable. Such incompleteness is easily modeled like subjective incompleteness. For instance, we may assume that $\rho_{\mathcal{L}}(\mathtt{inrole}(p, r, \tau)) = \mathtt{uu}$ for all p, r, τ.

- Past incompleteness: An audit log may not record the *existence* of certain relevant states, even those in the past. This has implications for enforcing temporal operators, e.g., we may be unable to check that $\boxminus\alpha$ simply because we cannot determine what states existed in the past. This form of incompleteness can be formally modeled by assuming that if a time point τ does not occur in an audit log $\mathcal{L}$, then $\rho_{\mathcal{L}}(\mathtt{in}(\tau, \tau', \tau'')) = \mathtt{uu}$. In the special case where it is *certain* that the time point τ *does not exist*, we would have $\rho_{\mathcal{L}}(\mathtt{in}(\tau, \tau', \tau'')) = \mathtt{ff}$.

Our enforcement algorithm (Section 4) works with partial structures in general and, hence, takes into account all these forms of incompleteness. We comment on some specific instances in Section 5.

Structure Extension In practice, system logs evolve over time by gathering more information. This leads to a natural order, $\mathcal{L}_1 \geq \mathcal{L}_2$ on structures ($\mathcal{L}_1$ *extends* $\mathcal{L}_2$), meaning that $\mathcal{L}_1$ has more information than $\mathcal{L}_2$. Formally, $\mathcal{L}_1 \geq \mathcal{L}_2$ iff $\mathrm{D}_{\mathcal{L}_1} \supseteq \mathrm{D}_{\mathcal{L}_2}$ and for all ground atoms P, $\rho_{\mathcal{L}_2}(P) \in \{\mathtt{tt}, \mathtt{ff}\}$ implies $\rho_{\mathcal{L}_1}(P) = \rho_{\mathcal{L}_2}(P)$. Thus, as structures extend, the valuation of an atom may change from `uu` to either `tt` or `ff`, but cannot change once it is either `tt` or `ff`. The following property follows by induction on φ:

- Monotonicity: $\mathcal{L}_1 \geq \mathcal{L}_2$ and $\mathcal{L}_2 \models \varphi$ imply $\mathcal{L}_1 \models \varphi$.

Replacing φ with $\overline{\varphi}$, we also obtain that $\mathcal{L}_1 \geq \mathcal{L}_2$ and $\mathcal{L}_2 \models \overline{\varphi}$ imply $\mathcal{L}_1 \models \overline{\varphi}$. Hence, if $\mathcal{L}_1 \geq \mathcal{L}_2$ then $\mathcal{L}_1$ preserves both the $\mathcal{L}_2$-truth and $\mathcal{L}_2$-falsity of every formula φ.

In the next section, we use this order between structures to both explain and prove formal properties of our enforcement algorithm.

4 Policy Enforcement

Our main technical contribution is an iterative process for enforcing policies written in the sublogic. Through the translation of Section 2.2, the same process applies to policies written in the entire policy logic. At each iteration, our algorithm takes as input a policy φ and the available audit log abstracted as a partial structure $\mathcal{L}$, and outputs a residual policy ψ that contains exactly the parts of φ that could not be verified due to lack of information in $\mathcal{L}$. Such an iteration is written $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$. In practice, ψ may contain subjective predicates and future obligations. Once more information becomes available, extending $\mathcal{L}$ to $\mathcal{L}'$ ($\mathcal{L}' \geq \mathcal{L}$), another iteration of the algorithm can be used with inputs ψ and $\mathcal{L}'$ to obtain a new formula ψ'. This process can be continued till a formula trivially equivalent to $\top$ or $\bot$ is obtained, or the truth or falsity of the remaining formula is decided by human intervention. By design, our algorithm satisfies three important properties:

- Termination: Each iteration terminates.
- Correctness: If $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$, then for all extensions $\mathcal{L}'$ of $\mathcal{L}$, $\mathcal{L}' \models \varphi$ iff $\mathcal{L}' \models \psi$.
- Minimality: If $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$, then an atom occurs in ψ only if it occurs in φ and its valuation on $\mathcal{L}$ is $\mathtt{uu}$.

The technically difficult part of the algorithm is its treatment of quantifiers $\forall x.\varphi$ and $\exists x.\varphi$ in the input. Indeed, for *propositional logic* (logic without quantifiers), an algorithm satisfying the three properties above can be constructed trivially: define $\mathsf{reduce}(\mathcal{L}, \varphi)$ to be the formula obtained by replacing each atom P in φ with $\top$ if $\rho_{\mathcal{L}}(P) = \mathtt{tt}$, with $\bot$ if $\rho_{\mathcal{L}}(P) = \mathtt{ff}$, and with P itself if $\rho_{\mathcal{L}}(P) = \mathtt{uu}$. This algorithm terminates because formulas are finite, its correctness can be proved by a simple induction on φ, and minimality is obvious from the definition of reduce.

However, as the reader may already anticipate, this simple idea does not extend to quantifiers. Consider, for instance, the behavior of the algorithm on inputs $\forall x.\varphi$ and $\mathcal{L}$. Because the output must be minimal, in order to reduce $\forall x.\varphi$, the algorithm must instantiate x with each possible element of the domain $\mathrm{D}_{\mathcal{L}}$ and check the truth or falsity of φ for that instance on $\mathcal{L}$. This immediately leads to non-termination because in models of realistic privacy policies the domain $\mathrm{D}_{\mathcal{L}}$ must be infinite, e.g., permissible time points and transmitted messages (which may contain free-text in natural language) are both infinite sets.

Given the need for an infinite domain, something intrinsic in φ must *limit* the number of *relevant instances* of x that need to be checked to a finite number. This is precisely what our restricted form of universal quantification, $\forall \vec{x}.(c \supset \varphi)$, accomplishes. Through syntactic restrictions of Figure 1 and other static checks described later, we ensure that there are only a finite number of instances of $\vec{x}$ for which c is true on the given structure $\mathcal{L}$. Further, all such instances can be mechanically computed from $\mathcal{L}$. Although fulfilling these requirements is non-trivial, given that they hold, the rest of the algorithm is natural and syntax-directed.

Briefly, our enforcement regime contains the following components:

- An efficiently checkable relation $\vdash \varphi$ on policies, called a *mode analysis* (borrowing the term from logic programming [4]), which ensures that the relevant instances of each quantified variable in φ are finite and computable.
- A function $\widehat{\mathtt{sat}}(\mathcal{L}, c)$ that computes all satisfying instances of the restriction c.
- The function $\mathsf{reduce}(\mathcal{L}, \varphi)$ that codifies a single iteration of enforcement. The definition of $\mathsf{reduce}(\mathcal{L}, \varphi)$ relies on $\widehat{\mathtt{sat}}(\mathcal{L}, c)$ and assumes that $\vdash \varphi$.

In the following, we explain each of these three components, starting with the main algorithm reduce (Section 4.1). After proving its correctness and minimality (Section 4.2), we proceed to define $\widehat{\mathtt{sat}}$ and the relation $\vdash \varphi$ (Section 4.3).

4.1 Iterative Enforcement Algorithm

The core of our enforcement regime is a computable function $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$, that discharges obligations from the prevalent policy φ using information from the extant structure $\mathcal{L}$ to obtain a residual policy ψ. Given an initial policy φ_0 and a sequence of structures $\mathcal{L}_1 \leq \mathcal{L}_2 \leq \ldots \leq \mathcal{L}_n$, the reduction algorithm can be applied repeatedly to obtain $\varphi_1, \ldots, \varphi_n$ such that $\mathsf{reduce}(\mathcal{L}_i, \varphi_{i-1}) = \varphi_i$. We write this process in symbols as $\varphi_0 \xrightarrow{\mathcal{L}_1} \varphi_1 \ldots \xrightarrow{\mathcal{L}_n} \varphi_n$. Correctness (Theorem 4.2) guarantees that φ_n is *equivalent* to φ_0 in all extensions of $\mathcal{L}_n$, while minimality (Theorem 4.3) certifies that φ_n contains only those atoms of φ_0 that could not be discharged using the information in $\mathcal{L}_n$ (by definition, $\mathcal{L}_n$ subsumes the information in $\mathcal{L}_1, \ldots, \mathcal{L}_{n-1}$). We note that our correctness and minimality results are independent of the frequency or scheme used for application of reduce.

The definition of $\mathsf{reduce}(\mathcal{L}, \varphi)$ has two dependencies, whose formal definitions are postponed to Section 4.3. First, the function assumes that its input φ is *well-moded*, formally written $\vdash \varphi$. Well-modedness is a static check, linear in the size of φ, which *ensures* that the satisfying instances of each restriction c in each quantifier in φ are *finite and computable*. Second, $\mathsf{reduce}(\mathcal{L}, \varphi)$ assumes a function $\widehat{\mathtt{sat}}(\mathcal{L}, c)$ that computes all satisfying instances of restriction c in structure $\mathcal{L}$. The output of $\widehat{\mathtt{sat}}(\mathcal{L}, c)$ is a finite set of substitutions $\{\sigma_1, \ldots, \sigma_n\}$, where each substitution σ_i is a finite map from free variables of c to ground terms. $\widehat{\mathtt{sat}}(\mathcal{L}, c)$ satisfies the following condition: $\mathcal{L} \models c\sigma$ iff $\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$.

The function $\mathsf{reduce}(\mathcal{L}, \varphi)$ is defined by induction on φ in Figure 2. For atoms P, $\mathsf{reduce}(\mathcal{L}, P)$ equals $\top$, $\bot$, or P, according to whether $\rho_{\mathcal{L}}(P)$ equals $\mathtt{tt}$, $\mathtt{ff}$, or $\mathtt{uu}$. In particular, in the absence of human input $\rho_{\mathcal{L}}(P_S) = \mathtt{uu}$ for a subjective atom P_S and hence, in the absence of human input, $\mathsf{reduce}(\mathcal{L}, P_S) = P_S$. The clauses for the connectives $\top$, $\bot$, $\wedge$, and $\vee$ are straightforward. To evaluate $\mathsf{reduce}(\mathcal{L}, \forall \vec{x}.(c \supset \varphi))$, we first determine the set of instances of $\vec{x}$ that satisfy c by calling $\widehat{\mathtt{sat}}(\mathcal{L}, c)$. For each such instance $\vec{t_1}, \ldots, \vec{t_n}$, we reduce $\varphi[\vec{t_i}/\vec{x}]$ to ψ_i through a recursive call to reduce. Because all instances of φ must hold in order for $\forall \vec{x}.(c \supset \varphi)$ to be true, the output is $\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'$, where the last conjunct ψ' records the fact that instances of $\vec{x}$ *other than* $\vec{t_1}, \ldots, \vec{t_n}$ have not been considered. The latter is necessary because there may be instances of $\vec{x}$ satisfying c in extensions of $\mathcal{L}$, but not $\mathcal{L}$ itself. Precisely, we define $S = \{\vec{t_1}, \ldots, \vec{t_n}\}$ and $\psi' = \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi)$. The new conjunct $\vec{x} \notin S$ prevents the instances $\vec{t_1}, \ldots, \vec{t_n}$ from being checked again in subsequent iterations. Formally, $\vec{x} \notin S$ is an objective predicate that encodes the negation of usual finite-set membership. The treatment of $\exists \vec{x}.(c \wedge \varphi)$ is dual; in that case, the output contains disjunctions because the truth of any one instance of φ suffices for the formula to hold.

Example 4.1. We illustrate iterative enforcement on the policy $\varphi_0 = \mathbf{G}\, \alpha_{pol2}$ that we obtained via translation in Example 2.3. The policy requires that the recipient of a request for information respond within 30 days with the information. We advise the reader to revisit the example for the definition of φ_0. For the purpose of explanation, let us define $\varphi(\tau, p, t)$ by pattern matching to be the formula satisfying $\varphi_0 = \forall \tau, p, t.\ (\mathtt{in}(\tau, 0, \infty) \wedge \mathtt{req}(p, t, \tau)) \supset \varphi(\tau, p, t)$. Informally, $\varphi(\tau, p, t)$ is the obligation that must be satisfied if principal p requests information about attribute t from her record at time τ.

Suppose that we first run $\mathsf{reduce}(\mathcal{L}, \varphi_0)$ in a structure $\mathcal{L}$ which has the states $1, 3, 7$, only one request — Alice's request for her medical record (attribute mr) at time 3, and no other information. Intuitively, this information implies that $\widehat{\mathtt{sat}}(\mathcal{L}, \mathtt{in}(\tau, 0, \infty) \wedge \mathtt{req}(p, t, \tau)) = \{(\tau, p, t) \mapsto (3, \text{Alice}, mr)\}$. (We check formally in Example 4.6 that this is actually the case.) Hence, by the definition of reduce, we know that $\mathsf{reduce}(\mathcal{L}, \varphi_0) = \psi_1 \wedge \varphi_0'$, where $\psi_1 = \mathsf{reduce}(\mathcal{L}, \varphi[(3, \text{Alice}, mr)/(\tau, p, t)])$

$$\mathsf{reduce}(\mathcal{L}, P) \quad = \quad \begin{cases} \top & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{tt} \\ \bot & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{ff} \\ P & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{uu} \end{cases}$$

$$\begin{array}{lcl} \mathsf{reduce}(\mathcal{L}, \top) & = & \top \\ \mathsf{reduce}(\mathcal{L}, \bot) & = & \bot \\ \mathsf{reduce}(\mathcal{L}, \varphi_1 \wedge \varphi_2) & = & \mathsf{reduce}(\mathcal{L}, \varphi_1) \wedge \mathsf{reduce}(\mathcal{L}, \varphi_2) \\ \mathsf{reduce}(\mathcal{L}, \varphi_1 \vee \varphi_2) & = & \mathsf{reduce}(\mathcal{L}, \varphi_1) \vee \mathsf{reduce}(\mathcal{L}, \varphi_2) \end{array}$$

$$\begin{array}{lcl} \mathsf{reduce}(\mathcal{L}, \forall \vec{x}.(c \supset \varphi)) & = & \text{let} \\ & & \quad \{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c) \\ & & \quad \{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^{n} \\ & & \quad S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\} \\ & & \quad \{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi[\vec{t_i}/\vec{x}])\}_{i=1}^{n} \\ & & \quad \psi' \leftarrow \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi) \\ & & \text{return} \\ & & \quad \psi_1 \wedge \ldots \wedge \psi_n \wedge \psi' \end{array}$$

$$\begin{array}{lcl} \mathsf{reduce}(\mathcal{L}, \exists \vec{x}.(c \wedge \varphi)) & = & \text{let} \\ & & \quad \{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c) \\ & & \quad \{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^{n} \\ & & \quad S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\} \\ & & \quad \{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi[\vec{t_i}/\vec{x}])\}_{i=1}^{n} \\ & & \quad \psi' \leftarrow \exists \vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi) \\ & & \text{return} \\ & & \quad \psi_1 \vee \ldots \vee \psi_n \vee \psi' \end{array}$$

Figure 2: Definition of $\mathsf{reduce}(\mathcal{L}, \varphi)$

and $\varphi_0' = \forall \tau, p, t.\ (\mathtt{in}(\tau, 0, \infty) \wedge \mathtt{req}(p, t, \tau) \wedge (\tau, p, t) \notin \{(3, \text{Alice}, mr)\}) \supset \varphi(\tau, p, t)$. The reader may check that because the trace has no other information, $\psi_1 = \varphi[(3, \text{Alice}, mr)/(\tau, p, t)]$, so the output of the reduction is $\psi_1 \wedge \varphi_0'$. Expansion of the formula ψ_1 shows that it is precisely the obligation that the recipient respond to Alice with her medical record in 30 days. Call this entire output φ_1.

Consider a second round of audit on the reduced policy φ_1 and an extended trace $\mathcal{L}'$ which has the additional state 11 in which Bob, in role "records", responds with a message M to Alice. Since $\varphi_1 = \psi_1 \wedge \varphi_0'$, we have $\mathsf{reduce}(\mathcal{L}', \varphi_1) = \mathsf{reduce}(\mathcal{L}', \psi_1) \wedge \mathsf{reduce}(\mathcal{L}', \varphi_0')$. The reader may check that $\mathsf{reduce}(\mathcal{L}', \varphi_0') = \varphi_0'$ because the top-level restriction in φ_0' has no satisfying instance in $\mathcal{L}'$. Thus, we consider here the reduction of ψ_1. Note that ψ_1 has the form $\exists \tau', q, m.\ ((\mathtt{in}(\tau', 3, 33) \wedge \mathtt{inrole}(q, \text{records}, \tau') \wedge \mathtt{send}(q, \text{Alice}, m, \tau')) \wedge \varphi'(\tau', q, m))$. To calculate its reduction, we first observe that from the information in $\mathcal{L}'$, it should follow that $\widehat{\mathtt{sat}}(\mathcal{L}', \mathtt{in}(\tau', 3, 33) \wedge \mathtt{inrole}(q, \text{records}, \tau') \wedge \mathtt{send}(q, \text{Alice}, m, \tau')) = \{(\tau', q, m) \mapsto (11, \text{Bob}, M)\}$. (Again, we check formally in Example 4.6 that this is the case.) Consequently, $\mathsf{reduce}(\mathcal{L}', \psi_1) = \psi_1' \vee \varphi_1'$, where $\psi_1' = \mathsf{reduce}(\mathcal{L}', \varphi'(11, \text{Bob}, M))$ and $\varphi_1' = \exists \tau', q, m.\ ((\mathtt{in}(\tau', 3, 33) \wedge \mathtt{inrole}(q, \text{records}, \tau') \wedge \mathtt{send}(q, \text{Alice}, m, \tau') \wedge (\tau', q, m) \notin \{(11, \text{Bob}, M)\}) \wedge \varphi'(\tau', q, m))$. We calculate ψ_1' below. The second disjunct φ_1' simply means that the policy is satisfied if at some point other than 11 (but before 33), someone in role "records" sends Alice's mr to her.

What is $\psi_1' = \mathsf{reduce}(\mathcal{L}', \varphi'(11, \text{Bob}, M))$? Expanding φ', we have $\varphi'(11, \text{Bob}, M) = \mathtt{contains}(M, \text{Alice}, mr, 11) \wedge \psi_2'$, where $\psi_2' = \forall \tau''. \ (\mathtt{in}(\tau'', 3, 11) \wedge \tau'' \neq 11) \supset \overline{\mathtt{ftr}}(\text{Alice}, mr, \tau'')$. Because $\mathtt{contains}$ is a subjective predicate, $\rho_{\mathcal{L}'}(\mathtt{contains}(M, \text{Alice}, mr, 11)) = \mathtt{uu}$ so, by definition, $\mathsf{reduce}(\mathcal{L}', \mathtt{contains}(M, \text{Alice}, mr, 11)) = \mathtt{contains}(M, \text{Alice}, mr, 11)$. Hence, if $\mathsf{reduce}(\mathcal{L}', \psi_2') = \psi_2''$, then $\psi_1' = \mathtt{contains}(M, \text{Alice}, mr, 11) \wedge \psi_2''$.

To compute ψ_2'', we note that $\widehat{\mathtt{sat}}(\mathcal{L}', \mathtt{in}(\tau'', 3, 11) \wedge \tau'' \neq 11) = \{\tau'' \mapsto 3, \tau'' \mapsto 7\}$. It follows that $\mathsf{reduce}(\mathcal{L}', \psi_2') = \psi_2'' = \overline{\mathtt{ftr}}(\text{Alice}, mr, 3) \wedge \overline{\mathtt{ftr}}(\text{Alice}, mr, 7) \wedge \psi_2'''$, where $\psi_2''' = \forall \tau''. \ (\mathtt{in}(\tau'', 3, 11) \wedge \tau'' \neq 11 \wedge \tau'' \notin \{3, 7\}) \supset \overline{\mathtt{ftr}}(\text{Alice}, mr, \tau'')$. Informally, ψ_2'' means that it should have been infeasible to respond to Alice at times 3 and 7 (which are the only two observed time points on $\mathcal{L}'$ before the response at time 11), and also at any other time points between 3 and 11 that may show up in extensions of $\mathcal{L}'$.

Putting back the various formulae, we have $\mathsf{reduce}(\mathcal{L}', \varphi_1) = (\psi_1' \vee \varphi_1') \wedge \varphi_0'$, where $\psi_1' = \mathtt{contains}(M, \text{Alice}, mr, 11) \wedge \psi_2''$ means that the message M sent to Alice at time 11 contain her mr and that it be infeasible to respond earlier (ψ_2''), φ_1' allows for the possibility to satisfy Alice's request through another response before time 33, and φ_0' enforces the top-level policy on any other requests. This is exactly what we might expect from an informal analysis. Further, note that the reduction exposes the ground subjective atoms $\mathtt{contains}(M, \text{Alice}, mr, 11)$, $\overline{\mathtt{ftr}}(\text{Alice}, mr, 3)$ and $\overline{\mathtt{ftr}}(\text{Alice}, mr, 7)$ for a human auditor to inspect and discharge.

4.2 Correctness and Minimality of Enforcement

The function reduce is correct in the sense that its input and output formulas contain the same obligations. Formally, if $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$, then in *all extensions of* $\mathcal{L}$, φ is true iff ψ is true and φ is false iff ψ is false.

Theorem 4.2 (Correctness of reduce)**.** *If* $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$ *and* $\mathcal{L}' \geq \mathcal{L}$*, then (1)* $\mathcal{L}' \models \varphi$ *iff* $\mathcal{L}' \models \psi$ *and (2)* $\mathcal{L}' \models \overline{\varphi}$ *iff* $\mathcal{L}' \models \overline{\psi}$*.*

Proof. See Appendix B, Theorem B.5. □

The proof of this theorem relies on correctness of $\widehat{\mathtt{sat}}$, which we prove in the next subsection (Theorem 4.5). Correctness of iterative enforcement is an immediate corollary of Theorem 4.2. We can prove by induction on n that if $\varphi_0 \xrightarrow{\mathcal{L}_1} \varphi_1 \ldots \xrightarrow{\mathcal{L}_n} \varphi_n$, then for all extensions $\mathcal{L}' \geq \mathcal{L}_n$, $\mathcal{L}' \models \varphi_n$ iff $\mathcal{L}' \models \varphi_0$ and $\mathcal{L}' \models \overline{\varphi_n}$ iff $\mathcal{L}' \models \overline{\varphi_0}$.

Next, we wish to prove that if $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$ then ψ is minimal with respect to φ and $\mathcal{L}$, i.e., an atom occurs in ψ only if it occurs in φ and its interpretation in $\mathcal{L}$ is unknown. Unfortunately, owing to quantification, there is no standard definition of the set of atoms of a formula of first-order logic. In the following, we provide one natural definition of the atoms of a formula and characterize minimality with respect to it; other similar characterizations are possible. If $\vdash \varphi$, we define the set of atoms of a formula φ with respect to a structure $\mathcal{L}$ as follows.

$$\begin{array}{lcl}
\mathtt{atoms}(\mathcal{L}, P_S) & = & \{P_S\} \\
\mathtt{atoms}(\mathcal{L}, P_O) & = & \{P_O\} \\
\mathtt{atoms}(\mathcal{L}, \top) & = & \{\} \\
\mathtt{atoms}(\mathcal{L}, \bot) & = & \{\} \\
\mathtt{atoms}(\mathcal{L}, \varphi_1 \wedge \varphi_2) & = & \mathtt{atoms}(\mathcal{L}, \varphi_1) \cup \mathtt{atoms}(\mathcal{L}, \varphi_2) \\
\mathtt{atoms}(\mathcal{L}, \varphi_1 \vee \varphi_2) & = & \mathtt{atoms}(\mathcal{L}, \varphi_1) \cup \mathtt{atoms}(\mathcal{L}, \varphi_2) \\
\mathtt{atoms}(\mathcal{L}, \forall \vec{x}.(c \supset \varphi)) & = & \bigcup_{\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)} \mathtt{atoms}(\mathcal{L}, \varphi\sigma) \\
\mathtt{atoms}(\mathcal{L}, \exists \vec{x}.(c \wedge \varphi)) & = & \bigcup_{\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)} \mathtt{atoms}(\mathcal{L}, \varphi\sigma)
\end{array}$$

The following theorem characterizes minimality of reduce with respect to the above definition of atoms in a formula.

Theorem 4.3 (Minimality)**.** *Suppose* $\vdash \varphi$ *and* $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$. *Then* $\mathtt{atoms}(\mathcal{L}, \psi) \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap \{P \mid \rho_{\mathcal{L}}(P) = \mathtt{uu}\}$.

Proof. See Appendix B, Theorem B.12. □

Example 4.4. Revisiting Example 4.1, we check that the output produced by the second reduction satisfies Theorem 4.3. Recall that the second reduction is $\mathsf{reduce}(\mathcal{L}', \varphi_1) = (\psi_1' \vee \varphi_1') \wedge \varphi_0'$. φ_1' and φ_0' each have top-level quantifiers whose guards have no satisfying instances in $\mathcal{L}'$, so, by definition of $\mathtt{atoms}$, φ_1' and φ_0' have no atoms w.r.t. $\mathcal{L}'$. Thus we turn to ψ_1'. It is easy to check that $\mathtt{atoms}(\mathcal{L}', \psi_1')$ is the three element set $\{\mathtt{contains}(M, \text{Alice}, mr, 11), \overline{\mathtt{ftr}}(\text{Alice}, mr, 3), \overline{\mathtt{ftr}}(\text{Alice}, mr, 7)\}$. Further, from the analysis of Example 4.1, each of these three atoms also exist in $\mathtt{atoms}(\mathcal{L}', \varphi_1)$. Finally, each of the three atoms is subjective, so each has a valuation $\mathtt{uu}$ in $\mathcal{L}$.

4.3 Quantifier Instantiation and Mode Analysis

Having described our main enforcement function reduce, we turn to the mode analysis relation $\vdash \varphi$ and the function $\widehat{\mathtt{sat}}$ on which the definition of reduce relies. The rest of this paper can be understood without understanding this section, so the disinclined reader may choose to skip it.

Input and Output The objective of our mode analysis, as mentioned earlier, is to ensure that the set of satisfying instances of quantified variables $\vec{x}$ in a restriction c be both finite and computable. Our method of mode analysis is inspired by, and based on a similar technique in logic programming (see, e.g. [4]). The key observation in mode analysis is that, for many predicates, the set of all satisfying instances on any given structure can be computed finitely if arguments in certain positions are ground. The reason why instances can be computed may vary from predicate to predicate; we illustrate some such computations from prior examples.

1. Given a ground m, the set of q, t such that $\mathtt{tagged}(m, q, t, \tau)$ holds is finite and can be computed from m itself, as we assumed in Example 2.1. (Note that the last argument τ is an artifact of our translation and is irrelevant here.)

2. For an action predicate like $\mathtt{send}(p_1, p_2, m, \tau)$, we can compute all instances of p_1, p_2, m, τ for which $\mathtt{send}(p_1, p_2, m, \tau)$ holds simply by querying the given system log.

3. Given ground τ_2, τ_3, we can compute all τ_1 such that $\texttt{in}(\tau_1, \tau_2, \tau_3)$ by looking at the states in the given system log and selecting the subset that lie in the interval $[\tau_2, \tau_3]$.

4. Given ground r and τ, we can compute all principals p such that $\texttt{inrole}(p, r, \tau)$ by looking at the roles' database.

Note that in each of the cases 1–4, we require that certain argument positions be ground (e.g., m in 1 and τ_2, τ_3 in 3), and compute others (e.g., q, t in 1 and τ_1 in 3). We call these the *input* and *output* argument positions, respectively. Formally, we represent input and output positions by two *partial functions* I and O (input and output) from predicates to 2^{N}, which we assume are given to us. The functions are partial because satisfying instances of certain predicates, including all subjective predicates, are not computable. Following the earlier example, we could choose:

1. $I(\texttt{tagged}) = \{1\}$, $O(\texttt{tagged}) = \{2, 3\}$

2. $I(\texttt{send}) = \{\}$, $O(\texttt{send}) = \{1, 2, 3, 4\}$

3. $I(\texttt{in}) = \{2, 3\}$, $O(\texttt{in}) = \{1\}$

4. $I(\texttt{inrole}) = \{2, 3\}$, $O(\texttt{inrole}) = \{1\}$

For a subjective predicate p_S, $I(p_S)$ and $O(p_S)$ are undefined. The sets $I(p)$ and $O(p)$ are called a moding of predicate p. If $i \in I(p)$ ($i \in O(p)$), we say that the ith argument of p is in input (output) mode. Certain arguments may be in neither input nor output mode, e.g., argument 4 of the predicate $\texttt{tagged}$. Also, the same predicate may be moded in multiple ways. For example, both the assignments ($I(\texttt{send}) = \{\}$, $O(\texttt{send}) = \{1, 2, 3, 4\}$) and ($I(\texttt{send}) = \{1\}$, $O(\texttt{send}) = \{2, 3, 4\}$) are correct. However, it suffices to assume that each predicate has a unique moding, because we can use different names for predicates with the same interpretation but different modings.

Substitution Computation A substitution σ is a finite map from variables to ground terms. Say that a substitution σ' extends a substitution σ, written $\sigma' \geq \sigma$, if $\texttt{dom}(\sigma') \supseteq \texttt{dom}(\sigma)$ and for all $x \in \texttt{dom}(\sigma)$, $\sigma(x) = \sigma'(x)$. We abstract the computation of terms in output positions from terms in input positions as a *partial computable function* $\texttt{sat}$. The input of the function is a pair containing a structure and an atom; its output is a finite *set* of substitutions. The function $\texttt{sat}$ satisfies the following condition:

> Given a structure $\mathcal{L}$ and an atom $p(t_1, \ldots, t_n)$ such that for all $i \in I(p)$, t_i is ground, $\texttt{sat}(\mathcal{L}, p(t_1, \ldots, t_n))$ is the set of all substitutions for variables in $\bigcup_{i \in O(p)} t_i$ that have extensions σ such that $\mathcal{L} \models p(t_1, \ldots, t_n)\sigma$.

For example, if in structure $\mathcal{L}$, principal Charlie has doctors Alice and Bob at time τ, then $\texttt{sat}(\mathcal{L}, \texttt{inrole}(p, \texttt{doc}(\text{Charlie}), \tau))$ would be the two element set $\{p \mapsto \text{Alice}, p \mapsto \text{Bob}\}$. If the input arguments in atom P are not ground, then $\texttt{sat}(\mathcal{L}, P)$ may be undefined. For example, if either τ_2 or τ_3 is not ground, then $\texttt{sat}(\mathcal{L}, \texttt{in}(\tau_1, \tau_2, \tau_3))$ is undefined. Because subjective predicates are not computable, $\texttt{sat}(\mathcal{L}, P_S)$ is also undefined for every subjective atom P_S. In practice, the function $\texttt{sat}(\mathcal{L}, P)$ could be implemented through queries to the database that stores the audit log.

We lift the function $\texttt{sat}$ to the function $\widehat{\texttt{sat}}$ that computes satisfying instances of restrictions. The specification of the lifted function $\widehat{\texttt{sat}}(\mathcal{L}, c)$ is similar to that of $\texttt{sat}$: Given a partially ground restriction c, $\widehat{\texttt{sat}}(\mathcal{L}, c)$ is a finite set of substitutions characterizing all satisfying instances of c.

$$\begin{array}{lcl}
\widehat{\texttt{sat}}(\mathcal{L}, p_O(t_1, \ldots, t_n)) & = & \texttt{sat}(\mathcal{L}, p_O(t_1, \ldots, t_n)) \\
\widehat{\texttt{sat}}(\mathcal{L}, \top) & = & \{\bullet\} \\
\widehat{\texttt{sat}}(\mathcal{L}, \bot) & = & \{\} \\
\widehat{\texttt{sat}}(\mathcal{L}, c_1 \wedge c_2) & = & \bigcup_{\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)} \sigma + \widehat{\texttt{sat}}(\mathcal{L}, c_2\sigma) \\
\widehat{\texttt{sat}}(\mathcal{L}, c_1 \vee c_2) & = & \widehat{\texttt{sat}}(\mathcal{L}, c_1) \cup \widehat{\texttt{sat}}(\mathcal{L}, c_2) \\
\widehat{\texttt{sat}}(\mathcal{L}, \exists x.c) & = & \widehat{\texttt{sat}}(\mathcal{L}, c) \backslash \{x\} \qquad (x \text{ fresh})
\end{array}$$

For atoms, the definition of $\widehat{\texttt{sat}}$ coincides with that of $\texttt{sat}$. Since $\top$ must always be true, $\widehat{\texttt{sat}}(\mathcal{L}, \top)$ contains only the empty substitution (denoted $\bullet$). Since $\bot$ can never be satisfied, $\widehat{\texttt{sat}}(\mathcal{L}, \bot)$ is empty. For $c_1 \wedge c_2$, the set of satisfying instances is obtained by taking those of c_1 (denoted σ above), and conjoining those with satisfying instances of $c_2\sigma$ (the operation $+$ is composition of substitutions with disjoint domains). The set of satisfying instances of $c_1 \vee c_2$ is the union of the satisfying instances of c_1 and c_2. Satisfying instances of $\exists x.c$ are obtained by taking those of c, and removing the substitutions for x.

$\widehat{\texttt{sat}}$ is a partial function because the underlying function $\texttt{sat}$ is partial. For instance, taking an example from Section 2, $\widehat{\texttt{sat}}(\mathcal{L}, \texttt{send}(p_1, p_2, m, \tau) \wedge \texttt{tagged}(m', q, t, \tau'))$ is undefined if m' is a variable because any substitution σ in the output of the recursive call $\widehat{\texttt{sat}}(\mathcal{L}, \texttt{send}(p_1, p_2, m, \tau))$ will not contain m' in its domain and, therefore, in the call to $\widehat{\texttt{sat}}(\mathcal{L}, \texttt{tagged}(m', q, t, \tau')\sigma)$, the first argument to $\texttt{tagged}$ will be non-ground. Since $I(\texttt{tagged}) = \{1\}$, this recursive call may fail to return an answer. On the other hand, $\widehat{\texttt{sat}}(\mathcal{L}, \texttt{send}(p_1, p_2, m, \tau) \wedge \texttt{tagged}(m, q, t, \tau'))$ is defined because the first argument of $\texttt{tagged}$ in the second recursive call is m, which is grounded by the substitution σ of the first recursive call. Despite being partial, $\widehat{\texttt{sat}}(\mathcal{L}, c)$ represents all satisfying instances of c, whenever it is defined, as formalized by the following theorem.

Theorem 4.5 (Correctness of $\widehat{\texttt{sat}}$)**.** *If $\widehat{\texttt{sat}}(\mathcal{L}, c)$ is defined then for any substitution σ' with $\texttt{dom}(\sigma') \supseteq \texttt{fv}(c)$, $\mathcal{L} \models c\sigma'$ iff there is a substitution $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c)$ such that $\sigma' \geq \sigma$.*

Proof. See Appendix B, Theorem B.3. □

Example 4.6. In Example 4.1, we informally evaluated $\widehat{\texttt{sat}}$ at several places. Here, we justify the first two evaluations. In the first instance, we said that $\widehat{\texttt{sat}}(\mathcal{L}, \texttt{in}(\tau, 0, \infty) \wedge \texttt{req}(p, t, \tau)) = \{(\tau, p, t) \mapsto (3, \text{Alice}, mr)\}$. This follows from the observation that from the information in the structure $\mathcal{L}$, we must have $\texttt{sat}(\mathcal{L}, \texttt{in}(\tau, 0, \infty)) = \{\tau \mapsto 1, \tau \mapsto 3, \tau \mapsto 7\}$, $\texttt{sat}(\mathcal{L}, \texttt{req}(p, t, 3)) = \{(p, t) \mapsto (\text{Alice}, mr)\}$ and $\texttt{sat}(\mathcal{L}, \texttt{req}(p, t, \tau)) = \{\}$ for $\tau \neq 3$. The result of applying $\widehat{\texttt{sat}}$ follows from its definition.

Similarly, we calculated that $\widehat{\texttt{sat}}(\mathcal{L}', \texttt{in}(\tau', 3, 33) \wedge \texttt{inrole}(q, \text{records}, \tau') \wedge \texttt{send}(q, \text{Alice}, m, \tau'))$ $= \{(\tau', q, m) \mapsto (11, \text{Bob}, M)\}$. This follows because, from the description of $\mathcal{L}'$, $\texttt{sat}(\mathcal{L}', \texttt{in}(\tau', 3, 33))$ $= \{\tau' \mapsto 3, \tau' \mapsto 7, \tau' \mapsto 11\}$, $\texttt{sat}(\mathcal{L}', \texttt{inrole}(q, \text{records}, T)) = \{q \mapsto \text{Bob}\}$ for $T = 11$ and $\{\}$ otherwise, and $\texttt{sat}(\mathcal{L}', \texttt{send}(q, p, m, \tau')) = \{(q, p, m, \tau') \mapsto (\text{Bob}, \text{Alice}, M, 11)\}$.

Mode Analysis Next, we define a static check of restrictions to rule out those on which $\widehat{\texttt{sat}}$ is not defined, e.g., $\texttt{send}(p_1, p_2, m, \tau) \wedge \texttt{tagged}(m', q, t, \tau')$ described earlier. This static check is what we call the mode analysis. A restriction that passes the check is called *well-moded*. Formally, we define well-modedness as a relation $\chi_I \vdash c : \chi_O$, where χ_I and χ_O are sets of variables. If the

$$\boxed{\chi_I \vdash c : \chi_O}$$

$$\frac{\forall k \in I(p_O).\ \mathtt{fv}(t_k) \subseteq \chi_I \qquad \chi_O = \chi_I \cup (\bigcup_{j \in O(p_O)} \mathtt{fv}(t_j))}{\chi_I \vdash p_O(t_1, \ldots, t_n) : \chi_O} \qquad \frac{}{\chi_I \vdash \top : \chi_I} \qquad \frac{}{\chi_I \vdash \bot : \chi_I}$$

$$\frac{\chi_I \vdash c_1 : \chi \qquad \chi \vdash c_2 : \chi_O}{\chi_I \vdash c_1 \wedge c_2 : \chi_O} \qquad \frac{\chi_I \vdash c_1 : \chi_1 \qquad \chi_I \vdash c_2 : \chi_2}{\chi_I \vdash c_1 \vee c_2 : \chi_1 \cap \chi_2} \qquad \frac{\chi_I \vdash c : \chi_O}{\chi_I \vdash \exists x.c : \chi_O \backslash \{x\}}$$

$$\boxed{\chi \vdash \varphi}$$

$$\frac{\forall k.\ \mathtt{fv}(t_k) \subseteq \chi}{\chi \vdash p(t_1, \ldots, t_k)} \qquad \frac{}{\chi \vdash \top} \qquad \frac{}{\chi \vdash \bot} \qquad \frac{\chi \vdash \varphi_1 \qquad \chi \vdash \varphi_2}{\chi \vdash \varphi_1 \wedge \varphi_2} \qquad \frac{\chi \vdash \varphi_1 \qquad \chi \vdash \varphi_2}{\chi \vdash \varphi_1 \vee \varphi_2}$$

$$\frac{\chi \vdash c : \chi_O \qquad \vec{x} \subseteq \chi_O \qquad \mathtt{fv}(c) \subseteq \chi \cup \vec{x} \qquad \chi_O \vdash \varphi}{\chi \vdash \forall \vec{x}.(c \supset \varphi)}$$

$$\frac{\chi \vdash c : \chi_O \qquad \vec{x} \subseteq \chi_O \qquad \mathtt{fv}(c) \subseteq \chi \cup \vec{x} \qquad \chi_O \vdash \varphi}{\chi \vdash \exists \vec{x}.(c \wedge \varphi)}$$

(In the rules for quantifiers, bound variables x or $\vec{x}$ must be renamed so that they are fresh.)

Figure 3: Moding Rules

relation holds, then for any σ with $\mathtt{dom}(\sigma) \supseteq \chi_I$ and any $\mathcal{L}$, $\widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$ is defined and, further, any substitution in it contains all of $\chi_O \backslash \chi_I$ in its domain. (χ_I and χ_O are analogues of inputs and outputs for restrictions.)

The relation $\chi_I \vdash c : \chi_O$ is defined by the rules of Figure 3, which also constitute a linear-time decision procedure for deciding the relation (with inputs c and χ_I and output χ_O). We explain some of the rules. An atom $p(t_1, \ldots, t_k)$ is well-moded if the free variables (abbreviated $\mathtt{fv}$) of input positions are ground (premise $\forall k \in I(p_O).\ \mathtt{fv}(t_k) \subseteq \chi_I$ of the first rule) and the output χ_O equals χ_I (which is already ground) unioned with $\bigcup_{j \in O(p_O)} \mathtt{fv}(t_j)$ (all of which must be in the domain of $\mathtt{sat}(\mathcal{L}, p(t_1, \ldots, t_n))$). The rule for conjunctions $c_1 \wedge c_2$ chains the outputs χ of c_1 into the inputs of c_2. The following theorem establishes that $\widehat{\mathtt{sat}}$ is total on well-moded restrictions and also establishes the relation between χ_I, χ_O and the substitutions in the output of $\widehat{\mathtt{sat}}$.

Theorem 4.7 (Totality of $\widehat{\mathtt{sat}}$)**.** *If $\chi_I \vdash c : \chi_O$, then for all structures $\mathcal{L}$ and all substitutions σ with $\mathtt{dom}(\sigma) \supseteq \chi_I$, $\widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$ is defined and, further, for each substitution $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_O$.*

Proof. See Appendix B, Theorem B.6. □

We extend the mode-check on restrictions to formulas φ of the sublogic. The objective of this mode-check is two-fold. First, the check ensures that all restrictions occurring in φ are well-moded in the sense described above. Second, for quantifiers $\forall \vec{x}.(c \supset \varphi')$ and $\exists \vec{x}.(c \wedge \varphi')$, the check ensures that the quantified variables $\vec{x}$ are contained in the *outputs* (χ_O) of the restriction c. (Hence, by

Theorems 4.5 and 4.7, any substitution in $\widehat{\mathtt{sat}}(\mathcal{L}, c)$ grounds $\vec{x}$, which is central to the termination of reduce.) The mode-check is formalized as the relation $\chi \vdash \varphi$, meaning that for any substitution σ with $\mathtt{dom}(\sigma) \supseteq \chi$, the formula $\varphi\sigma$ is well-moded. Its straightforward rules are shown in Figure 3. The rules constitute a linear-time decision procedure for checking the relation (with inputs χ and φ). In the rules for $\forall\vec{x}.(c \supset \varphi')$ and $\exists\vec{x}.(c \wedge \varphi')$, the first premises check that c is well-moded. The second premises ensure that the variables $\vec{x}$ are contained in the output χ_O of the mode check on c. The third premises ensure that c is closed. It can easily be checked that if $\chi \vdash \varphi$, then $\mathtt{fv}(\varphi) \subseteq \chi$.

We call a formula φ well-moded if $\{\} \vdash \varphi$, which we abbreviate to $\vdash \varphi$. The following theorem shows that on well-moded formulas, the function reduce is total. Further on a well-moded input, the output is also well-moded (so the output can used as input in a subsequent iteration).

Theorem 4.8 (Totality of reduce)**.** *If $\vdash \varphi$ then there is a ψ such that $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$ and $\vdash \psi$.*

Proof. See Appendix B, Theorem B.10. □

Example 4.9. It can easily be checked that the formulas $\mathbf{G}\,\alpha_{pol1}$ and $\mathbf{G}\,\alpha_{pol2}$ defined in Example 2.3 are all well-moded (e.g., $\vdash \mathbf{G}\,\alpha_{pol1}$) using the definitions of I and O presented at the beginning of this subsection.

5 Specific Instances of Enforcement

We analyze the behavior of our enforcement algorithm on two restricted classes of structures. First, we consider *objectively-complete* structures – those that map every objective atom to either $\mathtt{tt}$ or $\mathtt{ff}$ (Section 5.1). We show that for such structures $\mathcal{L}$, the output of $\mathsf{reduce}(\mathcal{L}, \varphi)$ can be simplified to conjunctions and disjunctions of ground subjective atoms through trivial rewriting (e.g., replacing $\top \wedge \psi$ with ψ), thus making it more amenable to human inspection. We also obtain a decision procedure to decide the truth and falsity of input formulas without subjective predicates.

Second, we consider *past-complete* structures, those that have complete information up to a specific point of time (Section 5.2). This corresponds to the standard assumption in every existing work on enforcement of temporal properties that the audit log contains all past information. In particular, we show that on past-complete traces, our algorithm yields a method to find violations of safety properties [2] and satisfactions of co-safety properties [11] at the earliest.

5.1 Execution on Objectively-Complete Structures

We analyze the output of $\mathsf{reduce}(\mathcal{L}, \varphi)$ when $\mathcal{L}$ is objectively-complete. Although objective-completeness requires that truth and falsity of objective atoms be determined even in the future, it may model some realistic settings. For instance, after audit-relevant information has been gathered from all possible sources, it may be assumed that any fact not explicitly seen is, by default, false. The resulting structure would be objectively-complete. Objectively-complete structures correspond to the case of subjective incompleteness from Section 3.

Definition 5.1. A structure $\mathcal{L}$ is called objectively-complete if for all objective atoms P_O, $\rho_{\mathcal{L}}(P_O) \in \{\mathtt{tt}, \mathtt{ff}\}$.

If a structure $\mathcal{L}$ is objectively-complete, then during the execution of $\mathsf{reduce}(\mathcal{L}, \varphi)$, *all* relevant substitutions can be found for quantifiers and *all* objective atoms can be replaced with either $\top$

or $\bot$. Indeed, we show in this subsection that if $\mathcal{L}$ is objectively-complete, then the output, ψ, of $\mathsf{reduce}(\mathcal{L}, \varphi)$ can be *rewritten* (using straightforward rewrite rules) to a logically equivalent formula that is either $\top$ or $\bot$ or contains only subjective atoms, conjunctions and disjunctions. This has practical importance because, as compared to a formula with quantifiers, a formula containing only subjective atoms, conjunctions and disjunctions is more amenable to human inspection and audit.

There are two kinds of rewriting we need to perform on the output ψ to reduce it to our desired form. First, we need to eliminate unnecessary occurrences of $\top$ and $\bot$ that arise either from occurrences of $\top$ and $\bot$ in the input formula, or as replacements of atoms that evaluate to `tt` and `ff` respectively. Such occurrences can be eliminated by repeatedly applying the following eight rewriting rules anywhere in the output:

$$\begin{array}{ll} \psi \wedge \top \to \psi & \qquad\qquad \top \wedge \psi \to \psi \\ \psi \wedge \bot \to \bot & \qquad\qquad \bot \wedge \psi \to \bot \\ \psi \vee \top \to \top & \qquad\qquad \top \vee \psi \to \top \\ \psi \vee \bot \to \psi & \qquad\qquad \bot \vee \psi \to \psi \end{array}$$

For example, if $\varphi = P_O \wedge P_S$ for an objective atom P_O and a subjective atom P_S and $\rho_{\mathcal{L}}(P_O) = \mathtt{tt}$, then $\mathsf{reduce}(\mathcal{L}, \varphi) = \top \wedge P_S$. This can be simplified to P_S using the second rule above. Note that each rule above preserves logical equivalence of formulas.

Second, we need to eliminate those quantified subformulas in the output that are called ψ' in the definition of reduce (Figure 2). These have the forms $\forall \vec{x}.((c \wedge x \notin S) \supset \varphi)$ and $\exists \vec{x}.((c \wedge x \notin S) \wedge \varphi)$. Because S contains all instances of $\vec{x}$ that satisfy c, $(c \wedge x \notin S)$ has no satisfying instances in $\mathcal{L}$, i.e., $\widehat{\mathsf{sat}}(\mathcal{L}, (c \wedge x \notin S)) = \{\}$. Further, because $\mathcal{L}$ is objectively-complete, any extension $\mathcal{L}'$ of $\mathcal{L}$ must agree with $\mathcal{L}$ on valuation of objective atoms, so, by Theorem 4.5, $\widehat{\mathsf{sat}}(\mathcal{L}', (c \wedge x \notin S)) = \{\}$. Consequently, $\forall \vec{x}.((c \wedge x \notin S) \supset \varphi)$ is logically equivalent to $\top$ in all extensions of $\mathcal{L}$ and $\exists \vec{x}.((c \vee x \notin S) \supset \varphi)$ is logically equivalent to $\bot$ in all extensions of $\mathcal{L}$. This immediately yields the following two rules for elimination of quantifiers from the output of reduce.

$$\forall \vec{x}.(c \supset \varphi) \to \top \qquad\qquad \exists \vec{x}.(c \wedge \varphi) \to \bot$$

We point out that, unlike the eight rewriting rules presented earlier, the two rewriting rules above do not preserve logical equivalence in general, but they preserve logical equivalence when applied to the output $\psi = \mathsf{reduce}(\mathcal{L}, \varphi)$ for objectively-complete $\mathcal{L}$.

Let $\to^*$ denote the reflexive-transitive closure of $\to$. Since $\to$ makes formulas strictly smaller, it cannot be applied indefinitely to any formula. Further, even though a formula may be rewritten in many ways using a single application of $\to$, the formula obtained by applying $\to$ exhaustively starting from a fixed initial formula is unique because $\to$ is confluent.

Theorem 5.2. *Suppose $\mathcal{L}$ is objectively-complete, $\vdash \varphi$ and $\psi = \mathsf{reduce}(\mathcal{L}, \varphi)$. Then $\psi \to^* \psi'$, where (1) ψ' is either $\top$, or $\bot$, or contains only subjective atoms and the connectives $\wedge$, $\vee$, and (2) For all $\mathcal{L}' \geq \mathcal{L}$, $\mathcal{L}' \models \psi$ iff $\mathcal{L}' \models \psi'$ and $\mathcal{L}' \models \overline{\psi}$ iff $\mathcal{L}' \models \overline{\psi'}$.*

Proof. See Appendix C, Theorem C.4. □

An interesting special case arises on inputs φ without any subjective predicates. In this case, it can be proved by induction on φ that if $\mathcal{L}$ is objectively-complete, then either $\mathcal{L} \models \varphi$ or $\mathcal{L} \models \overline{\varphi}$ (either φ is true in $\mathcal{L}$ or it is false). Interestingly, for such inputs, Theorem 5.2 yields a *decision procedure* for determining the truth or falsity of φ in $\mathcal{L}$. The proof of this fact is straightforward.

By minimality of reduce (Theorem 4.3), the output ψ of $\mathsf{reduce}(\mathcal{L}, \varphi)$ cannot contain any subjective atoms if φ does not contain them, so neither can the formula ψ' obtained by rewriting in Theorem 5.2. Hence, ψ' must be either $\top$ or $\bot$. If $\psi' = \top$, then by Theorem 4.2, $\mathcal{L} \models \varphi$, and if $\psi' = \bot$, then by the same theorem, $\mathcal{L} \models \overline{\varphi}$. This is a decision procedure because both reduce and $\rightarrow^*$ terminate.

5.2 Execution on Past-Complete Structures

Next, we analyze our enforcement algorithm on structures that have complete information up to a specific point of time, say τ_0. We call such structures τ_0-past-complete or, briefly, τ_0-complete. Past-completeness corresponds to future incompleteness from Section 3 and is practically relevant because in many cases, audit logs record all relevant events as they happen and the *entire* history is available to an enforcement algorithm. In fact, this is a standard assumption in all existing literature on either runtime or post-hoc enforcement of temporal properties. The classic result in this context is that, under this assumption, a runtime monitor can detect both violation of so-called safety properties (a given bad event never happens) and satisfaction of so-called co-safety properties (a given good event happens at some time either in the past or in the future) at the earliest possible time. In the rest of this subsection, we show that on past-complete structures similar results hold for our enforcement method.

We start by formally defining past-complete structures, then adapt a standard characterization of safety and co-safety properties in temporal logic to our setting, and finally prove that the function reduce, together with rewriting $\rightarrow$, yields a method to enforce both safety and co-safety properties. It is important to mention here that violation or satisfaction of a property cannot be defined formally if the property has subjective predicates. Consequently, we assume in this subsection, like existing literature on the subject, that policies do not contain subjective predicates.

Definition 5.3. Given a ground time τ_0, a structure $\mathcal{L}$ is called τ_0-past-complete or τ_0-complete if the following two conditions hold:

1. For all predicates p, all ground $t_1, \ldots, t_n$ and all $\tau \leq \tau_0$, $\rho_{\mathcal{L}}(p(t_1, \ldots, t_n, \tau)) \in \{\mathtt{tt}, \mathtt{ff}\}$.

2. For all ground τ_1, τ_2, τ_3 such that $\tau_1 \leq \tau_0$, $\rho_{\mathcal{L}}(\mathtt{in}(\tau_1, \tau_2, \tau_3)) \in \{\mathtt{tt}, \mathtt{ff}\}$.

The first condition means that the truth or falsity of every atom in the temporal logic can be determined at time τ if $\tau \leq \tau_0$. The second condition states that $\mathcal{L}$ records all relevant states up to time τ_0.

Safety and Co-safety Informally, a safety property states that a specified bad condition is never satisfied. Dually, a co-safety property states that a specified good condition is satisfied at some time (either in the past or in the future). Although the two kinds of properties are often characterized in terms of traces (semantically) [2, 11], characterizations of the two kinds of properties as classes of formulas in logic are more relevant for us. It is known [23] that safety properties correspond to formulas of the form $\mathbf{G}\, \alpha_p$, where $\mathbf{G}$ is the "in every state" operator introduced in Example 2.3 and α_p is an arbitrary formula of the temporal logic not containing any future operators ($\square$ and U). In words, $\mathbf{G}\, \alpha_p$ means that in every state (the bad condition) $\neg \alpha_p$ does not hold. As an illustration, the policy $\mathbf{G}\, \alpha_{pol1}$ in Example 2.3 is a safety property, but $\mathbf{G}\, \alpha_{pol2}$ is not because it contains a future operator. Dually, co-safety properties can be characterized as formulas of the form

$\mathbf{F}\,\alpha_p = \exists\tau.(\mathtt{in}(\tau,0,\infty) \wedge (\alpha_p)^\tau)$, informally meaning that in some state τ, (the good condition) α_p holds.[1]

We say that a safety property $\mathbf{G}\,\alpha_p$ is *violated* at time τ in a structure $\mathcal{L}$ if $\mathcal{L} \models \overline{(\alpha_p)^\tau}$. In other words, $\mathbf{G}\,\alpha_p$ is violated at time τ if at that time, the negation of α_p holds in $\mathcal{L}$. Similarly, we say that a co-safety property $\mathbf{F}\,\alpha_p$ is *satisfied* at time τ in a structure $\mathcal{L}$ if $\mathcal{L} \models (\alpha_p)^\tau$.

Our first result (Theorem 5.4) is that if a safety property $\mathbf{G}\,\alpha_p$ is violated at time τ in a structure $\mathcal{L}$ that is τ_0-complete ($\tau \leq \tau_0$), then $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\,\alpha_p) \to^* \bot$ (and conversely). This result is important because it implies that violations of safety properties can be detected in the *next iteration of enforcement after they occur* if audit logs contain all past information. An analogous result – Theorem 5.5 – holds for co-safety properties, wherein satisfaction can be detected at the earliest. The justification for both theorems is similar to that for Theorem 5.2, but more involved. Because both reduce and $\to^*$ terminate, the theorems also provide decision procedures for enforcing safety and co-safety properties on past-complete structures.

Theorem 5.4 (Enforcement of safety properties)**.** *Suppose $\mathbf{G}\,\alpha_p$ is a safety property, $\vdash \mathbf{G}\,\alpha_p$, $\mathcal{L}$ is τ_0-complete, and for all τ, $(\rho_{\mathcal{L}}(\mathtt{in}(\tau,0,\infty)) = \mathtt{tt}) \Rightarrow \tau \leq \tau_0$. Then, $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\,\alpha_p) \to^* \bot$ iff there is a τ such that $\mathcal{L} \models \mathtt{in}(\tau,0,\tau_0)$ and $\mathcal{L} \models \overline{(\alpha_p)^\tau}$.*

Proof. See Appendix C, Theorem C.12. □

Theorem 5.5 (Enforcement of co-safety properties)**.** *Suppose $\mathbf{F}\,\alpha_p$ is a co-safety property, $\vdash \mathbf{F}\,\alpha_p$, $\mathcal{L}$ is τ_0-complete, and for all τ, $(\rho_{\mathcal{L}}(\mathtt{in}(\tau,0,\infty)) = \mathtt{tt}) \Rightarrow \tau \leq \tau_0$. Then, $\mathsf{reduce}(\mathcal{L}, \mathbf{F}\,\alpha_p) \to^* \top$ if and only if there is a τ such that $\mathcal{L} \models \mathtt{in}(\tau,0,\tau_0)$ and $\mathcal{L} \models (\alpha_p)^\tau$.*

Proof. See Appendix C, Theorem C.13. □

Example 5.6. We check Theorem 5.4 on the safety property $\mathbf{G}\,\alpha_{pol1}$ from Example 2.3. The policy states that if a message m is sent by p_1 to p_2 for purpose u and the message is tagged as containing q's data about attribute t (which is a form of *phi*), then either the recipient p_2 is q's doctor and the purpose u is treatment, or q has previously consented to this message transmission.

We consider a simple structure $\mathcal{L}$ in which this policy is violated. $\mathcal{L}$ has only one time point 7, at which principal A sends principal B a message M. The message M is labeled with purpose *test* ($\mathtt{purp\_in}(\mathit{test}, \mathit{treatment})$ holds) and tagged as containing principal C's information about attribute *meds* (medications), which is a form of *phi*. Further, B, the recipient, is not C's doctor. Suppose that we audit at a later point of time (10) and that $\mathcal{L}$ described above is 10-complete. Since there is no other information in $\mathcal{L}$ besides what has been mentioned, C has not consented explicitly to this message transmission, so the policy has been violated at time 7. We seek to verify that $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\,\alpha_{pol1}) \to^* \bot$.

We start by computing $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\,\alpha_{pol1})$. The reader is advised to revisit the definition of $\mathbf{G}\,\alpha_{pol1}$ in Example 2.3. At the top-level, $\mathbf{G}\,\alpha_{pol1}$ contains a universal quantifier with restriction $c = (\mathtt{in}(\tau,0,\infty) \wedge \mathtt{send}(p_1,p_2,m,\tau) \wedge \mathtt{purp}(m,u,\tau) \wedge \mathtt{tagged}(m,q,t,\tau) \wedge \mathtt{attr\_in}(t,\mathit{phi},\tau))$. Computing $\widehat{\mathtt{sat}}(\mathcal{L}, c)$ yields $\{(\tau,p_1,p_2,m,u,q,t) \mapsto (7, \mathrm{A}, \mathrm{B}, M, \mathit{test}, \mathrm{C}, \mathit{meds})\}$. Hence, $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\,\alpha_{pol1})$ $= \mathsf{reduce}(\mathcal{L}, \varphi_1) \wedge \varphi_0'$, where φ_1 is shown below and φ_0' is almost a copy of the original policy, with a larger restriction. The only aspect of φ_0' relevant for this example is that it contains a top-level universal quantifier.

[1] We have not seen this characterization of co-safety properties in literature, but it is easily derived as the dual of the known characterization of safety properties.

$$\begin{aligned}\varphi_1 = {} & (\mathtt{inrole}(\mathrm{B}, \mathtt{doc}(\mathrm{C}), 7) \wedge \\ & \quad \mathtt{purp\_in}(\mathit{test}, \mathit{treatment}, 7)) \vee \\ & (\exists \tau'.\ (\mathtt{in}(\tau', 0, 7) \wedge \\ & \qquad \mathtt{consents}(\mathrm{C}, \mathtt{sendaction}(\mathrm{A}, \mathrm{B}, (\mathrm{C}, \mathit{meds})), \tau')))\end{aligned}$$

Next, we calculate $\mathsf{reduce}(\mathcal{L}, \varphi_1)$. Since $\rho_{\mathcal{L}}(\mathtt{inrole}(\mathrm{B}, \mathtt{doc}(\mathrm{C}), 7)) = \mathtt{ff}$ and $\rho_{\mathcal{L}}(\mathtt{purp\_in}(\mathit{test}, \mathit{treatment}, 7)) = \mathtt{tt}$, $\mathsf{reduce}(\mathcal{L}, \varphi_1) = (\bot \wedge \top) \vee \mathsf{reduce}(\mathcal{L}, \varphi_2)$, where φ_2 is the second disjunct of φ_1. Finally, we compute $\mathsf{reduce}(\mathcal{L}, \varphi_2)$. The top-level connective of φ_2 is an existential quantifier restricted by $\mathtt{in}(\tau', 0, 7)$. Since $\widehat{\mathtt{sat}}(\mathcal{L}, \mathtt{in}(\tau', 0, 7)) = \{\tau' \mapsto 7\}$, $\mathsf{reduce}(\mathcal{L}, \varphi_2) = \mathsf{reduce}(\mathcal{L}, \varphi_3) \vee \varphi_2'$, where $\varphi_3 = \mathtt{consents}(\mathrm{C}, \mathtt{sendaction}(\mathrm{A}, \mathrm{B}, (\mathrm{C}, \mathit{meds})), 7)$ and φ_2' begins with an existential quantifier. Clearly, $\mathsf{reduce}(\mathcal{L}, \varphi_3) = \bot$. Putting the pieces back together, we get $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\, \alpha_{pol1}) = ((\bot \wedge \top) \vee (\bot \vee \varphi_2')) \wedge \varphi_0'$.

Since φ_0' and φ_2' begin with a universal and an existential quantifier, they can be rewritten to $\top$ and $\bot$ respectively. So, $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\, \alpha_{pol1}) \to ((\bot \wedge \top) \vee (\bot \vee \bot)) \wedge \top$, which can easily be rewritten to $\bot$, thus indicating a violation. If we change the example to avoid a violation, say by setting $\rho_{\mathcal{L}}(\mathtt{inrole}(\mathrm{B}, \mathtt{doc}(\mathrm{C}), 7))$ to $\mathtt{tt}$ instead of $\mathtt{ff}$, then the result of rewriting changes from $\bot$ to $\top$, indicating a lack of violation thus far. Finally, if we do not assume that $\mathcal{L}$ is past-complete, then the rewriting of φ_2' to $\bot$ is unsound because there may be an extension of $\mathcal{L}$ in which φ_2' is true and, hence, the original property may not have been violated, but our procedure would conclude that it is. So, past-completeness is a necessary assumption in Theorem 5.4 (and also Theorem 5.5).

6 Application to HIPAA

We comment on application of our algorithm to transmission-relevant clauses of the HIPAA Privacy Rule. These clauses can be viewed as a template for actual privacy policies, which may be obtained by instantiating abstract roles like "covered entity" in HIPAA with actual roles like "doctor", "nurse", etc. In prior work on PrivacyLFP [16], we have shown that all 84 transmission-related clauses in HIPAA can be represented in the logic. Since we have restricted the syntax of quantifiers in this paper to facilitate enforcement, an immediate question is whether we can still represent all the clauses of HIPAA in our logic. A careful re-analysis of the prior work reveals that 81 of the 84 clauses fall in the fragment considered in this paper. The three remaining clauses, namely Sections 164.506(c)(4), 164.512(k)(1)(i), and 164.512(k)(1)(iv) of HIPAA, contain quantifiers with subjective restrictions. However, in each such case, the formula under the quantifier contains only subjective predicates and, therefore, the entire formula may be considered a single subjective predicate. With this minor change, the algorithm of Section 4 can be applied to all 84 clauses of HIPAA.

The next question is the usefulness of the algorithm, given that HIPAA contains many subjective predicates (in fact, 578 out of a total of 881 atoms in our formalization of HIPAA are subjective). The answer to this question is two-fold. First, irrespective of the percentage of subjective atoms, one practical advantage of using our algorithm is that it instantiates quantifiers automatically using log data, which could otherwise be a daunting task for a human auditor.

Second, our algorithm automatically discharges objective atoms from fully instantiated formulas, leaving only subjective atoms for a human auditor. As discussed in the prior work, with a slight amount of design effort, e.g., standardizing message formats, 402 of the subjective atoms can be mechanized, leaving a total of 176 subjective atoms, and improving the effectiveness of the algorithm significantly. A reasonable method to quantify the effectiveness of the algorithm on instantiated formulas is to calculate the ratio of the number of objective atoms to the total number of atoms for all 84 clauses. (A more accurate assessment can be made if we also know how frequently each clause of HIPAA gets instantiated, but this is impossible without real data.) In Appendix D, we list for each clause the numbers of subjective and objective atoms in it (#S and #O respectively), as well as the number of subjective atoms that can be mechanized by simple design effort such as standardizing message formats (#O'). The ratio (#O' + #O) / (#S + #O) shown in the last column is an estimate of the percentage of the clause our algorithm will reduce automatically, assuming that the required design effort has been made. Based on these figures, we count that in 17 clauses, all atoms can be reduced automatically; in 24 other clauses, at least 80% of the atoms can be reduced automatically; and in 29 other clauses, at least 50% of the atoms can be reduced automatically. On the other hand, in 6 clauses our algorithm cannot reduce any atoms automatically but 5 out of these 6 clauses contain exactly one subjective atom each.

In summary, even though completely automatic enforcement of policies derived from HIPAA is impossible due its use of subjective predicates, our algorithm can help reduce the burden of human auditors significantly, both by instantiating quantifiers automatically and by discharging objective atoms in fully instantiated formulas.

7 Related Work

Policy Enforcement with Temporal Logic A lot of prior work addresses the problem of *runtime monitoring* of policies expressed in Linear Temporal Logic (LTL) [5, 7, 10, 28, 30, 31] and its extensions [7, 29, 30]. Although similar in the spirit of enforcing policies, the intended deployment of our work is different: we expect our algorithm to be used for after-the-fact audit for violations, rather than for online monitoring. Consequently, the issue of retaining only necessary portions of logs, which is central to runtime monitoring, is largely irrelevant for our work (and hence not considered in this paper).

Comparing only the expressiveness of the logic, our work is more advanced than all existing work on policy enforcement. First, we enforce a large fragment of first-order temporal logic, whereas prior work is either limited to propositional logic [5, 28, 31], or, when quantifiers are considered, they are severely restricted [7, 29, 30]. A recent exception to such syntactic restrictions is the work of Basin et al. [10], to which we compare in detail below. Second, no prior work considers either subjective predicates, or the possibility of gaps in past information, both of which our partial structures and enforcement algorithm account for.

Recent work by Basin et al. [10] considers runtime monitoring over an expressive fragment of Metric First-order Temporal Logic. Similar to our work, Basin et al. allow quantification over infinite domains, and use a form of mode analysis (called a safe-range analysis) to ensure finiteness during enforcement. However, Basin et al's mode analysis is weaker than ours; in particular, it cannot relate the same variable in the input and output positions of two different conjuncts of a restriction and requires that each free variable appear in at least one predicate with a finite model. As a consequence, some policies such as α_{pol1} (Example 2.1), whose top-level restriction $(\mathtt{send}(p_1, p_2, m) \wedge \mathtt{purp}(m, u) \wedge \ldots)$ contains a variable u not occurring in any predicate with a finite model, cannot be enforced in their framework, but can be enforced in ours. Due to their goal of runtime enforcement, Basin et al. use auxiliary data structures to cache relevant portions of the log in memory, which may form the basis of useful optimizations in an implementation of our work.

Cederquist et al. [14] present a proof-based system for a-posteriori audit, where policy obligations are discharged by constructing formal proofs. The leaves of proofs are established from logs, but the audit process only checks that an obligation has been satisfied somewhere in the past, thus allowing only for obligations of the form $\diamondminus\varphi$. Further, there is no systematic mechanism to instantiate quantifiers in proofs. However, using connectives of linear logic, the mechanism admits policies that rely on consumable permissions.

The idea of iteratively rewriting the policy over evolving audit logs has been considered previously [28, 31], but only for propositional logic. Bauer et al. [5] use a different approach for iterative enforcement: they convert an LTL formula with limited first-order quantification to a Büchi automaton and check whether the automaton accepts the input log. Further, they also use a three-valued semantic model similar to ours, but assume past-completeness. Three-valued structures have also been considered in work on generalized model checking [13, 19]. However, the problems addressed in that line of work are different; the objective there is to check whether there exist extensions of a given structure in which a formula is satisfied (or falsified).

Policy Specification Several variants of LTL have been used to *specify* the properties of programs, business processes and security and privacy policies [8, 9, 16, 18, 22]. Our representation of policies and our logic, PrivacyLFP, draw inspiration from LPU [8].

Further, several access-control models have extensions for specifying usage control and future obligations [12, 17, 20, 21, 25–27]. Some of these models assume a pre-defined notion of obligations [21, 25]. For instance, Irwin et al [21] model obligations as tuples containing the subject of the obligation, the actions to be performed, the objects that are targets of the actions and the time frames of the obligations. Other models leave specifications for obligations abstract [12, 20, 27]. Such specific models and the ensuing policies can be encoded in our logic using quantifiers and temporal operators.

There also has been much work on analyzing the properties of policies represented in formal models. For instance, Ni et al. study the interaction between obligation and authorization [25], Irwin et al. have analyzed accountability problems with obligations [21], and Dougherty et al. have modeled the interaction between obligations and programs [17]. These methods are orthogonal to our objective of policy enforcement. It may be possible to adapt ideas from these papers to analyze similar properties of policies expressed in PrivacyLFP also.

Finally, privacy languages such as EPAL [6] and privacyAPI [24] do not include obligations or temporal modalities as primitives, and are less expressive than our framework.

8 Conclusion

We have presented an expressive and provably correct iterative method for enforcing privacy policies that works by reducing policies, even in the face of incomplete system logs. Our method is expressive enough to enforce real privacy legislation like HIPAA, yet tractable due to a carefully designed static analysis. Under standard assumptions about system logs, we obtain methods to mechanically enforce safety and co-safety properties.

Our planned next step is to implement the proposed enforcement mechanism and to test its performance on real privacy legislation. A specific goal is to develop generic optimization and caching techniques that encompass all forms of log incompleteness, to the extent possible. Prior work on runtime monitoring may provide valuable insights in this regard, but a significant challenge is to generalize it beyond past-completeness.

References

[1] FairWarning®. http://www.fairwarningaudit.com.

[2] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

[3] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–203, 1994.

[4] Krzysztof R. Apt and Elena Marchiori. Reasoning about Prolog programs: From modes through types to assertions. *Formal Aspects of Computing*, 6(6):743–765, 1994.

[5] Franz Baader, Andreas Bauer, and Marcel Lippmann. Runtime verification using a temporal description logic. In *Proceedings of the 7th international conference on Frontiers of combining systems*, FroCoS'09, pages 149–164, 2009.

[6] Michael Backes, Birgit Pfitzmann, and Matthias Schunter. A toolkit for managing enterprise privacy policies. In *European Symposium on Research in Computer Security*, LNCS 2808, pages 101–119, 2003.

[7] Howard Barringer, Allen Goldberg, Klaus Havelund, and Koushik Sen. Rule-based runtime verification. In *Proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, pages 44–57, 2004.

[8] Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *Proceedings of the 27th IEEE Symposium on Security and Privacy*, pages 184–198, May 2006.

[9] David Basin, Felix Klaedtke, and Samuel Müller. Monitoring security policies with metric first-order temporal logic. In *Proceeding of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 23–34, 2010.

[10] David A. Basin, Felix Klaedtke, and Samuel Müller. Policy monitoring in first-order temporal logic. In *Proceedings of the 22nd International Conference on Computer Aided Verification (CAV)*, pages 1–18, 2010.

[11] Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 2010. To appear.

[12] Claudio Bettini, Sushil Jajodia, X. Sean Wang, and Duminda Wijesekera. Provisions and obligations in policy rule management. *Journal of Network and Systems Management*, 11:351–372, 2003.

[13] Glenn Bruns and Patrice Godefroid. Generalized model checking: Reasoning about partial state spaces. In *Proceedings of the 11th International Conference on Concurrency Theory*, CONCUR '00, pages 168–182, 2000.

[14] J. G. Cederquist, R. Corin, M. A. C. Dekker, S. Etalle, J. I. den Hartog, and G. Lenzini. Audit-based compliance control. *International Journal of Information Security*, 6(2):133–151, 2007.

[15] Deloitte & Touche and the Ponemon Institute. Enterprise@Risk: 2007 Privacy and Data Protection Survey. White Paper, December 2007.

[16] Henry DeYoung, Deepak Garg, Limin Jia, Dilsun Kaynar, and Anupam Datta. Experiences in the logical specification of the HIPAA and GLBA privacy laws. In *Proceedings of the 9th annual ACM Workshop on Privacy in the Electronic Society (WPES)*, 2010.

[17] Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Obligations and their interaction with programs. In *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS)*, pages 375–389, 2007.

[18] Christopher Giblin, Alice Y. Liu, Samuel Müller, Birgit Pfitzmann, and Xin Zhou. Regulations expressed as logical models (REALM). In *Proceeding of the 18th Annual Conference on Legal Knowledge and Information Systems (JURIX)*, pages 37–48, 2005.

[19] Patrice Godefroid and Michael Huth. Model checking vs. generalized model checking: Semantic minimizations for temporal logics. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, pages 158–167, 2005.

[20] Manuel Hilty, David A. Basin, and Alexander Pretschner. On obligations. In *Proceedings of the 10th European Symposium on Research in Computer Security*, pages 98–117, 2005.

[21] Keith Irwin, Ting Yu, and William H. Winsborough. On the modeling and analysis of obligations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 134–143, 2006.

[22] Y. Liu, S. Müller, and K. Xu. A static compliance-checking framework for business process models. *IBM Systems Journal*, 46:335–361, 2007.

[23] Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, 1995.

[24] Michael J. May, Carl A. Gunter, and Insup Lee. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In *Proceedings of the 19th IEEE Workshop on Computer Security Foundations (CSFW)*, pages 85–97, 2006.

[25] Qun Ni, Elisa Bertino, and Jorge Lobo. An obligation model bridging access control policies and privacy policies. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 133–142, 2008.

[26] OASIS XACML Committee. Extensible access control markup language (XACML) v2.0, 2004. Available at http://www.oasis-open.org/specs/#xacmlv2.0.

[27] Jaehong Park and Ravi Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 57–64, 2002.

[28] Grigore Roşu and Klaus Havelund. Rewriting-based techniques for runtime verification. *Automated Software Engineering*, 12:151–197, 2005.

[29] Muriel Roger and Jean Goubault-Larrecq. Log auditing through model-checking. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations (CSF)*, pages 220–236, 2001.

[30] Oleg Sokolsky, Usa Sammapun, Insup Lee, and Jesung Kim. Run-time checking of dynamic properties. *Electronic Notes in Theoretical Computer Science*, 144:91–108, 2006.

[31] Prasanna Thati and Grigore Roşu. Monitoring algorithms for metric temporal logic specifications. *Electronic Notes in Theoretical Computer Science*, 113:145–162, 2005.

[32] US Congress. Health Insurance Portability and Accountability Act of 1996, Privacy Rule. 45 CFR 164, 2002. Available at http://www.access.gpo.gov/nara/cfr/waisidx_07/45cfr164_07.html.

A Details from Section 2

The full definition of the $\overline{\varphi}$ is shown below:

$$
\begin{array}{ccc}
\overline{p_O(t_1,\ldots,t_n)} & = & \overline{p_O}(t_1,\ldots,t_n) \\
\overline{p_S(t_1,\ldots,t_n)} & = & \overline{p_S}(t_1,\ldots,t_n) \\
\overline{\top} & = & \bot \\
\overline{\bot} & = & \top \\
\overline{\varphi \wedge \psi} & = & \overline{\varphi} \vee \overline{\psi} \\
\overline{\varphi \vee \psi} & = & \overline{\varphi} \wedge \overline{\psi} \\
\overline{\forall \vec{x} \notin S.(c \supset \varphi)} & = & \exists \vec{x} \notin S.(c \wedge \overline{\varphi}) \\
\overline{\exists \vec{x} \notin S.(c \wedge \varphi)} & = & \forall \vec{x} \notin S.(c \supset \overline{\varphi})
\end{array}
$$

The full translation $(\bullet)^\tau$ from the temporal logic to the sublogic is shown below:

$$
\begin{array}{rcl}
(p_O(t_1,\ldots,t_n))^\tau & = & p_O(t_1,\ldots,t_n,\tau) \\
(\top)^\tau & = & \top \\
(\bot)^\tau & = & \bot \\
(c_1 \wedge c_2)^\tau & = & (c_1)^\tau \wedge (c_2)^\tau \\
(c_1 \vee c_2)^\tau & = & (c_1)^\tau \vee (c_2)^\tau \\
(\exists x.c)^\tau & = & \exists x.(c)^\tau
\end{array}
$$

$$
\begin{array}{rcl}
(p_O(t_1,\ldots,t_n))^\tau & = & p_O(t_1,\ldots,t_n,\tau) \\
(p_S(t_1,\ldots,t_n))^\tau & = & p_S(t_1,\ldots,t_n,\tau) \\
(\top)^\tau & = & \top \\
(\bot)^\tau & = & \bot \\
(\alpha \wedge \beta)^\tau & = & (\alpha)^\tau \wedge (\beta)^\tau \\
(\alpha \vee \beta)^\tau & = & (\alpha)^\tau \vee (\beta)^\tau \\
(\neg\alpha)^\tau & = & \overline{(\alpha)^\tau} \\
(\forall\vec{x}.(c \supset \alpha))^\tau & = & \forall\vec{x}.((c)^\tau \supset (\alpha)^\tau) \\
(\exists\vec{x}.(c \wedge \alpha))^\tau & = & \exists\vec{x}.((c)^\tau \wedge (\alpha)^\tau) \\
(\downarrow x.\alpha)^\tau & = & (\alpha[\tau/x])^\tau \\
(\alpha\,\mathrm{S}\,\beta)^\tau & = & \exists\tau'.(\mathtt{in}(\tau',0,\tau) \wedge (\beta)^{\tau'} \\
 & & \quad \wedge\, (\forall\tau''.((\mathtt{in}(\tau'',\tau',\tau) \wedge \tau' \neq \tau'') \\
 & & \qquad\qquad \supset (\alpha)^{\tau''}))) \\
(\alpha\,\mathrm{U}\,\beta)^\tau & = & \exists\tau'.(\mathtt{in}(\tau',\tau,\infty) \wedge (\beta)^{\tau'} \\
 & & \quad \wedge\, (\forall\tau''.((\mathtt{in}(\tau'',\tau,\tau') \wedge \tau'' \neq \tau') \\
 & & \qquad\qquad \supset (\alpha)^{\tau''}))) \\
(\boxminus\alpha)^\tau & = & \forall\tau'.(\mathtt{in}(\tau',0,\tau) \supset (\alpha)^{\tau'}) \\
(\Box\alpha)^\tau & = & \forall\tau'.(\mathtt{in}(\tau',\tau,\infty) \supset (\alpha)^{\tau'})
\end{array}
$$

B Proofs from Section 4

This appendix contains proofs of theorems presented in Section 4. The proofs are presented in an order different from the order of theorems in the main body of the paper because of dependencies in the proofs.

Lemma B.1 (Monotonicity)**.** *$\mathcal{L}' \geq \mathcal{L}$ and $\mathcal{L} \models \varphi$ imply $\mathcal{L}' \models \varphi$.*

Proof. By induction on φ. □

Lemma B.2 (Consistency)**.** *For all $\mathcal{L}$ and φ, either $\mathcal{L} \not\models \varphi$ or $\mathcal{L} \not\models \overline{\varphi}$.*

Proof. By induction on φ. □

Theorem B.3 (Correctness of $\widehat{\mathtt{sat}}$; Theorem 4.5)**.** *If $\widehat{\mathtt{sat}}(\mathcal{L}, c)$ is defined then for any substitution σ' with $\mathtt{dom}(\sigma') \supseteq \mathtt{fv}(c)$, $\mathcal{L} \models c\sigma'$ iff there is a substitution $\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$ such that $\sigma' \geq \sigma$.*

Proof. By induction on c and case analysis of its top-level constructor.

Case. $c = p_o(t_1, \ldots, t_n)$. Then, $\widehat{\texttt{sat}}(\mathcal{L}, c) = \texttt{sat}(\mathcal{L}, c)$. The result follows from the condition that $\texttt{sat}$ is required to satisfy (Section 4.3).

Case. $c = \top$. Then, $\widehat{\texttt{sat}}(\mathcal{L}, c) = \{\bullet\}$. If $\mathcal{L} \models c\sigma'$, σ' trivially extends $\bullet$ by definition. Conversely, any substitution σ' trivially satisfies $\mathcal{L} \models \top\sigma'$.

Case. $c = \bot$. Then, $\widehat{\texttt{sat}}(\mathcal{L}, c) = \{\}$. The result is vacuously true in both directions because $\mathcal{L} \not\models \bot\sigma'$, and $\sigma' \notin \{\}$.

Case. $c = c_1 \wedge c_2$. Then, $\widehat{\texttt{sat}}(\mathcal{L}, c) = \bigcup_{\sigma_1 \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)} \sigma_1 + \widehat{\texttt{sat}}(\mathcal{L}, c_2\sigma_1)$. Clearly, if this exists, then $\widehat{\texttt{sat}}(\mathcal{L}, c_1)$ must be defined also, and for each $\sigma_1 \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)$, $\widehat{\texttt{sat}}(\mathcal{L}, c_2\sigma_1)$ must also be defined.

Suppose $\mathcal{L} \models (c_1 \wedge c_2)\sigma'$. By definition of $\models$, we get $\mathcal{L} \models c_1\sigma'$ and $\mathcal{L} \models c_2\sigma'$. By the i.h., the former implies that there is a $\sigma_1 \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)$ such that $\sigma' \geq \sigma_1$. This also implies that $c_2\sigma' = (c_2\sigma_1)\sigma'$. So, $\mathcal{L} \models c_2\sigma'$ implies $\mathcal{L} \models (c_2\sigma_1)\sigma'$. Consequently, by the i.h. on $c_2\sigma_1$, there must be a $\sigma_2 \in \widehat{\texttt{sat}}(\mathcal{L}, c_2\sigma_1)$ such that $\sigma' \geq \sigma_2$. It follows that $\sigma' \geq \sigma_1 + \sigma_2$. Clearly, $(\sigma_1 + \sigma_2) \in (\sigma_1 + \widehat{\texttt{sat}}(\mathcal{L}, c_2\sigma_1)) \subseteq (\bigcup_{\sigma_1 \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)} \sigma_1 + \widehat{\texttt{sat}}(\mathcal{L}, c_2\sigma_1)) = \widehat{\texttt{sat}}(\mathcal{L}, c)$, as required.

Conversely, suppose that there is a $\sigma \in \bigcup_{\sigma_1 \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)} \sigma_1 + \widehat{\texttt{sat}}(\mathcal{L}, c_2\sigma_1)$ and $\sigma' \geq \sigma$ with $\texttt{dom}(\sigma') \supseteq \texttt{fv}(\sigma)$. We need to show that $\mathcal{L} \models (c_1 \wedge c_2)\sigma'$ or, equivalently, $\mathcal{L} \models c_1\sigma'$ and $\mathcal{L} \models c_2\sigma'$. By set-theory, there must be a $\sigma_1 \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)$ and a $\sigma_2 \in \widehat{\texttt{sat}}(\mathcal{L}, c_2\sigma_1)$ such that $\sigma = \sigma_1 + \sigma_2$. Clearly, $\sigma' \geq \sigma_1$. So, by the i.h., we immediately have $\mathcal{L} \models c_1\sigma'$. Similarly, $\sigma' \geq \sigma_2$. So, by i.h. on $c_2\sigma_1$, $\mathcal{L} \models c_2\sigma_1\sigma'$. But, $c_2\sigma_1\sigma' = c_2\sigma'$. Therefore, $\mathcal{L} \models c_2\sigma'$.

Case. $c = c_1 \vee c_2$. Then, $\widehat{\texttt{sat}}(\mathcal{L}, c) = \widehat{\texttt{sat}}(\mathcal{L}, c_1) \cup \widehat{\texttt{sat}}(\mathcal{L}, c_2)$. If this is defined, then, clearly, both $\widehat{\texttt{sat}}(\mathcal{L}, c_1)$ and $\widehat{\texttt{sat}}(\mathcal{L}, c_2)$ must be defined.

Suppose $\mathcal{L} \models (c_1 \vee c_2)\sigma'$. By definition of $\models$, we get that either $\mathcal{L} \models c_1\sigma'$ or $\mathcal{L} \models c_2\sigma'$. We consider here the former case (the latter is similar). So $\mathcal{L} \models c_1\sigma'$. By the i.h., there is a $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)$ such that $\sigma' \geq \sigma_1$. The proof is complete by noting that $\sigma_1 \in \widehat{\texttt{sat}}(\mathcal{L}, c_1) \in \widehat{\texttt{sat}}(\mathcal{L}, c)$.

Conversely, suppose that there is a $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c_1) \cup \widehat{\texttt{sat}}(\mathcal{L}, c_2)$ and $\sigma' \geq \sigma$ with $\texttt{dom}(\sigma') \supseteq \texttt{fv}(\sigma)$. We need to show that $\mathcal{L} \models (c_1 \vee c_2)\sigma'$ or, equivalently, either $\mathcal{L} \models c_1\sigma'$ or $\mathcal{L} \models c_2\sigma'$. From $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c_1) \cup \widehat{\texttt{sat}}(\mathcal{L}, c_2)$, we get that either $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)$ or $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c_2)$. Consider the former case (the latter is similar): $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c_1)$. By i.h. on c_1, we immediately get $\mathcal{L} \models c_1\sigma'$, as required.

Case. $c = \exists x.c'$. Then, $\widehat{\texttt{sat}}(\mathcal{L}, c) = \widehat{\texttt{sat}}(\mathcal{L}, c')\backslash\{x\}$. If this is defined, then, clearly, $\widehat{\texttt{sat}}(\mathcal{L}, c')$ must also be defined.

Suppose $\mathcal{L} \models (\exists x.c')\sigma'$. By definition of $\models$, there must be a t such that $\mathcal{L} \models c'[t/x]\sigma'$. By i.h. on c', there must be a $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c')$ such that $(\sigma' + [x \mapsto t]) \geq \sigma$. Clearly, $\sigma' \geq \sigma\backslash\{x\}$ and $\sigma\backslash\{x\} \in \widehat{\texttt{sat}}(\mathcal{L}, c)$, as required.

Conversely, suppose that there is a $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c')\backslash\{x\}$ and $\sigma' \geq \sigma$ with $\texttt{dom}(\sigma') \supseteq \texttt{fv}(c)$. We need to show that $\mathcal{L} \models c\sigma'$. Because $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c')\backslash\{x\}$, there is a $\sigma'' \in \widehat{\texttt{sat}}(\mathcal{L}, c')$ and a t such that $\sigma'' = \sigma + [x \mapsto t]$. Clearly, $\sigma' + [x \mapsto t] \geq \sigma + [x \mapsto t] = \sigma''$. By i.h. on c', $\mathcal{L} \models c'[t/x]\sigma'$, which implies (by definition of $\models$) that $\mathcal{L} \models (\exists x.c')\sigma'$, i.e., $\mathcal{L} \models c\sigma'$. □

Lemma B.4 (Duality of reduce). $\mathsf{reduce}(\mathcal{L}, \overline{\varphi}) = \overline{\mathsf{reduce}(\mathcal{L}, \varphi)}$.

Proof. By a straightforward induction on φ. We show some representative cases below.

Case. $\varphi = P$. Then,

$$\mathsf{reduce}(\mathcal{L}, P) \;=\; \begin{cases} \top & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{tt} \\ \bot & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{ff} \\ P & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{uu} \end{cases}$$

We consider all three possible subcases on $\rho_{\mathcal{L}}(P)$. If $\rho_{\mathcal{L}}(P) = \mathtt{tt}$, then, by definition, $\rho_{\mathcal{L}}(\overline{P}) = \mathtt{ff}$, so $\mathsf{reduce}(\mathcal{L}, \overline{P}) = \bot = \overline{\top} = \overline{\mathsf{reduce}(\mathcal{L}, P)}$. The case of $\rho_{\mathcal{L}}(P) = \mathtt{ff}$ is similar. For $\rho_{\mathcal{L}}(P) = \mathtt{uu}$, we have $\rho_{\mathcal{L}}(\overline{P}) = \mathtt{uu}$, so $\mathsf{reduce}(\mathcal{L}, \overline{P}) = \overline{P} = \overline{\mathsf{reduce}(\mathcal{L}, P)}$.

Case. $\varphi = \forall \vec{x}.(c \supset \varphi')$. Then, $\mathsf{reduce}(\mathcal{L}, \varphi)$ is calculated as follows:

```
reduce(L, ∀x⃗.(c ⊃ φ')) = let
                              $\{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c)$
                              $\{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^{n}$
                              $S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\}$
                              $\{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])\}_{i=1}^{n}$
                              $\psi' \leftarrow \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$
                          return
                              $\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'$
```

Note that $\overline{\varphi} = \exists \vec{x}.(c \wedge \overline{\varphi'})$. Consequently, $\mathsf{reduce}(\mathcal{L}, \overline{\varphi})$ is calculated as follows, where we have renamed some bound variables to distinguish them from those in the above display.

```
reduce(L, ∃x⃗.(c ∧ φ'‾)) = let
                              $\{\sigma'_1, \ldots, \sigma'_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c)$
                              $\{\vec{t'_i} \leftarrow \sigma'_i(\vec{x})\}_{i=1}^{n}$
                              $S' \leftarrow \{\vec{t'_1}, \ldots, \vec{t'_n}\}$
                              $\{\psi'_i \leftarrow \mathsf{reduce}(\mathcal{L}, \overline{\varphi'}[\vec{t'_i}/\vec{x}])\}_{i=1}^{n}$
                              $\psi'' \leftarrow \exists \vec{x}.((c \wedge \vec{x} \notin S') \wedge \overline{\varphi'})$
                          return
                              $\psi'_1 \vee \ldots \vee \psi'_n \vee \psi''$
```

We must have $\sigma_i = \sigma'_i$ (because both are calculated using $\widehat{\mathtt{sat}}(\mathcal{L}, c)$) and, consequently, $\vec{t_i} = \vec{t'_i}$ and $S = S'$. Thus, by the i.h., we get that $\mathsf{reduce}(\mathcal{L}, \overline{\varphi'}[\vec{t'_i}/\vec{x}]) = \overline{\mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])}$, i.e., $\psi'_i = \overline{\psi_i}$. Also observe that directly from definition of duality, $\psi'' = \overline{\psi'}$. Thus, $\mathsf{reduce}(\mathcal{L}, \overline{\varphi}) = \psi'_1 \vee \ldots \vee \psi'_n \vee \psi'' = \overline{\psi_1} \vee \ldots \vee \overline{\psi_n} \vee \overline{\psi'} = \overline{\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'} = \overline{\mathsf{reduce}(\mathcal{L}, \varphi)}$. □

Theorem B.5 (Correctness of reduce; Theorem 4.2)**.** *If $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$ and $\mathcal{L}' \geq \mathcal{L}$, then (1) $\mathcal{L}' \models \varphi$ iff $\mathcal{L}' \models \psi$ and (2) $\mathcal{L}' \models \overline{\varphi}$ iff $\mathcal{L}' \models \overline{\psi}$.*

Proof. First observe that (1) implies (2). Why? Suppose (1) holds for all φ. We need to show that (2) holds. So suppose $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$ and $\mathcal{L}' \geq \mathcal{L}$. By Lemma B.4, $\mathsf{reduce}(\mathcal{L}, \overline{\varphi}) = \overline{\psi}$. Applying the assumed (1) to $\overline{\varphi}$ instead of φ, we immediately deduce that $\mathcal{L}' \models \overline{\varphi}$ iff $\mathcal{L}' \models \overline{\psi}$, as required.

Hence, we only need to prove (1). We do that by induction on φ, and a case analysis of its top-level constructor.

Case. $\varphi = P$. Then,

$$\mathsf{reduce}(\mathcal{L}, \varphi) \;=\; \begin{cases} \top & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{tt} \\ \bot & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{ff} \\ P & \text{if } \rho_{\mathcal{L}}(P) = \mathtt{uu} \end{cases}$$

We consider three subcases on the value of $\rho_{\mathcal{L}}(P)$.

Subcase. $\rho_{\mathcal{L}}(P) = \mathtt{tt}$. Here, $\psi = \top$. First, assume that $\mathcal{L}' \models \varphi$. Then, we need to prove that $\mathcal{L}' \models \psi$, i.e, $\mathcal{L}' \models \top$. This follows directly from the definition of $\models$. Conversely, assume that $\mathcal{L}' \models \psi$. We need to prove that $\mathcal{L}' \models P$. By definition, this is equivalent to proving $\rho_{\mathcal{L}'}(P) = \mathtt{tt}$, which follows immediately from the subcase assumption $\rho_{\mathcal{L}}(P) = \mathtt{tt}$ and the assumption $\mathcal{L}' \geq \mathcal{L}$.

Subcase. $\rho_{\mathcal{L}}(P) = \mathtt{ff}$. Here $\psi = \bot$. First, assume that $\mathcal{L}' \models \varphi$. We need to show that $\mathcal{L}' \models \psi$. From the subcase assumption, we have $\rho_{\mathcal{L}}(P) = \mathtt{ff}$, so the definition of $\mathcal{L}' \geq \mathcal{L}$ implies that $\rho_{\mathcal{L}'}(P) = \mathtt{ff}$. However, $\mathcal{L}' \models \varphi$ implies $\mathcal{L}' \models P$, i.e., $\rho_{\mathcal{L}'}(P) = \mathtt{tt}$ – a contradiction. Thus, $\mathcal{L}' \models \psi$ holds vacuously.

Conversely, suppose that $\mathcal{L}' \models \psi$, i.e., $\mathcal{L}' \models \bot$. By definition of $\models$, this is a contradiction, so $\mathcal{L}' \models \varphi$ holds vacuously, as required.

Subcase. $\rho_{\mathcal{L}}(P) = \mathtt{uu}$. Here, $\varphi = \psi = P$, so the case is trivial.

Case. $\varphi = \top$. Then, $\psi = \mathsf{reduce}(\mathcal{L}, \varphi) = \mathsf{reduce}(\mathcal{L}, \top) = \top$. Since $\varphi = \psi$, the case is trivial.

Case. $\varphi = \bot$. Then, $\psi = \mathsf{reduce}(\mathcal{L}, \varphi) = \mathsf{reduce}(\mathcal{L}, \bot) = \bot$. Since $\varphi = \psi$, the case is trivial.

Case. $\varphi = \varphi_1 \wedge \varphi_2$. Then, $\psi = \mathsf{reduce}(\mathcal{L}, \varphi_1) \wedge \mathsf{reduce}(\mathcal{L}, \varphi_2)$, so both the conjuncts exist. First, suppose that $\mathcal{L}' \models \varphi$, i.e., $\mathcal{L}' \models \varphi_1$ and $\mathcal{L}' \models \varphi_2$. By the i.h., $\mathcal{L}' \models \mathsf{reduce}(\mathcal{L}, \varphi_1)$ and $\mathcal{L}' \models \mathsf{reduce}(\mathcal{L}, \varphi_2)$ or, equivalently, $\mathcal{L}' \models \psi$.

Conversely, suppose that $\mathcal{L}' \models \psi$. Then, $\mathcal{L}' \models \mathsf{reduce}(\mathcal{L}, \varphi_1)$ and $\mathcal{L}' \models \mathsf{reduce}(\mathcal{L}, \varphi_2)$. By the i.h., $\mathcal{L}' \models \varphi_1$ and $\mathcal{L}' \models \varphi_2$, i.e., $\mathcal{L}' \models \varphi$.

Case. $\varphi = \varphi_1 \vee \varphi_2$. Then, $\psi = \mathsf{reduce}(\mathcal{L}, \varphi_1) \vee \mathsf{reduce}(\mathcal{L}, \varphi_2)$, so both the disjuncts exist. First, suppose that $\mathcal{L}' \models \varphi$, i.e., either $\mathcal{L}' \models \varphi_1$ or $\mathcal{L}' \models \varphi_2$. By the i.h., either $\mathcal{L}' \models \mathsf{reduce}(\mathcal{L}, \varphi_1)$ or $\mathcal{L}' \models \mathsf{reduce}(\mathcal{L}, \varphi_2)$. Equivalently, $\mathcal{L}' \models \psi$.

Conversely, suppose that $\mathcal{L}' \models \psi$. Then, either $\mathcal{L}' \models \mathsf{reduce}(\mathcal{L}, \varphi_1)$ or $\mathcal{L}' \models \mathsf{reduce}(\mathcal{L}, \varphi_2)$. By the i.h., either $\mathcal{L}' \models \varphi_1$ or $\mathcal{L}' \models \varphi_2$. Equivalently, $\mathcal{L}' \models \varphi$.

Case. $\varphi = \forall \vec{x}.(c \supset \varphi')$. Then, $\psi = \mathsf{reduce}(\mathcal{L}, \varphi)$ is calculated as follows.

```
$\mathsf{reduce}(\mathcal{L}, \forall \vec{x}.(c \supset \varphi'))$ = let
        $\{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c)$
        $\{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^{n}$
        $S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\}$
        $\{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])\}_{i=1}^{n}$
        $\psi' \leftarrow \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$
    return
        $\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'$
```

So $\psi = \psi_1 \wedge \ldots \wedge \psi_n \wedge \forall\vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$. First, suppose that $\mathcal{L}' \models \varphi$, i.e., $\mathcal{L}' \models \forall\vec{x}.(c \supset \varphi')$. We need to prove that $\mathcal{L}' \models \psi$, i.e., $\mathcal{L}' \models \psi_i$ and $\mathcal{L}' \models \forall\vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$. We first prove that $\mathcal{L}' \models \psi_i$. Because $\mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}]) = \psi_i$, by the i.h., it suffices to show that $\mathcal{L}' \models \varphi'[\vec{t_i}/\vec{x}]$. From the definition of $\mathcal{L}' \models \forall\vec{x}.(c \supset \varphi')$, either $\mathcal{L}' \models \overline{c}[\vec{t_i}/\vec{x}]$ or $\mathcal{L}' \models \varphi'[\vec{t_i}/\vec{x}]$. Hence, it suffices to prove that $\mathcal{L}' \not\models \overline{c}[\vec{t_i}/\vec{x}]$. Suppose, for the sake of contradiction, that $\mathcal{L}' \models \overline{c}[\vec{t_i}/\vec{x}]$. Since $\sigma_i \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$, Theorem B.3 yields $\mathcal{L} \models c\sigma_i$, i.e., $\mathcal{L} \models c[\vec{t_i}/\vec{x}]$ (note that because $\forall\vec{x}.(c \supset \varphi')$ is closed, $\mathtt{fv}(c) \subseteq \vec{x}$; so $c[\vec{t_i}/\vec{x}] = c\sigma_i$). Hence, by Lemma B.1, $\mathcal{L}' \models c[\vec{t_i}/\vec{x}]$, which, by Lemma B.2, contradicts the earlier fact $\mathcal{L}' \models \overline{c}[\vec{t_i}/\vec{x}]$.

Next, we show that $\mathcal{L}' \models \forall\vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$. Following the definition of $\models$, pick any $\vec{t}$. We show that either $\mathcal{L}' \models \overline{(c \wedge \vec{x} \notin S)}[\vec{t}/\vec{x}]$ or $\mathcal{L}' \models \varphi'[\vec{t}/\vec{x}]$. Since we assumed that $\mathcal{L}' \models \forall\vec{x}.(c \supset \varphi')$, either $\mathcal{L}' \models \overline{c}[\vec{t}/\vec{x}]$ or $\mathcal{L}' \models \varphi'[\vec{t}/\vec{x}]$. The proof is complete by observing that $\mathcal{L}' \models \overline{c}[\vec{t}/\vec{x}]$ implies $\mathcal{L}' \models \overline{(c \wedge \vec{x} \notin S)}[\vec{t}/\vec{x}]$.

Conversely, assume that $\mathcal{L}' \models \psi$, i.e., $\mathcal{L}' \models \psi_i$ and $\mathcal{L}' \models \forall\vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$. We need to prove that $\mathcal{L}' \models \varphi$, i.e., $\mathcal{L}' \models \forall\vec{x}.(c \supset \varphi')$. Following the definition of $\models$, pick any $\vec{t}$. We need to prove that either $\mathcal{L}' \models \overline{c}[\vec{t}/\vec{x}]$ or $\mathcal{L}' \models \varphi'[\vec{t}/\vec{x}]$. We consider two subcases. Either $\vec{t} \in S$ or $\vec{t} \notin S$.

Subcase. $\vec{t} \in S$. Then, $\vec{t} = \vec{t_i}$ for some i. Since $\mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}]) = \psi_i$ and $\mathcal{L}' \models \psi_i'$, by the i.h. we get $\mathcal{L}' \models \varphi'[\vec{t_i}/\vec{x}]$, as required.

Subcase. $\vec{t} \notin S$. We already know that $\mathcal{L}' \models \forall\vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$. So, either $\mathcal{L}' \models \overline{(c \wedge \vec{x} \notin S)}[\vec{t}/\vec{x}]$ or $\mathcal{L}' \models \varphi'[\vec{t}/\vec{x}]$. If the latter, we are done, so assume the former. Thus, $\mathcal{L}' \models \overline{(c \wedge \vec{x} \notin S)}[\vec{t}/\vec{x}]$, i.e., $\mathcal{L}' \models \overline{c}[\vec{t}/\vec{x}] \vee \vec{t} \in S$. This immediately implies that either $\mathcal{L}' \models \overline{c}[\vec{t}/\vec{x}]$ or $\vec{t} \in S$. The former case is sufficient for our purpose, and the latter case contradicts the subcase assumption.

Case. $\varphi = \exists\vec{x}.(c \wedge \varphi')$. Then, $\mathsf{reduce}(\mathcal{L}, \varphi)$ is calculated as follows.

```
$\mathsf{reduce}(\mathcal{L}, \exists\vec{x}.(c \wedge \varphi'))$ = let
    $\{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c)$
    $\{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^n$
    $S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\}$
    $\{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])\}_{i=1}^n$
    $\psi' \leftarrow \exists\vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')$
  return
    $\psi_1 \vee \ldots \vee \psi_n \vee \psi'$
```

So, $\psi = \psi_1 \vee \ldots \vee \psi_n \vee \exists\vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')$. First suppose that $\mathcal{L}' \models \varphi$. We show that $\mathcal{L}' \models \psi$. Following the definition of $\models$ on $\mathcal{L}' \models \varphi$, we obtain a $\vec{t}$ such that $\mathcal{L}' \models c[\vec{t}/\vec{x}]$ and $\mathcal{L}' \models \varphi'[\vec{t}/\vec{x}]$. We consider two subcases: either $\vec{t} \in S$ or $\vec{t} \notin S$.

Subcase. $\vec{t} \in S$. So, $\vec{t} = \vec{t_i}$ for some i and from $\mathcal{L}' \models \varphi'[\vec{t}/\vec{x}]$ we obtain $\mathcal{L}' \models \varphi'[\vec{t_i}/\vec{x}]$. Since $\mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}]) = \psi_i$, by the i.h., we get $\mathcal{L}' \models \psi_i$, which immediately implies $\mathcal{L}' \models \psi$.

Subcase. $\vec{t} \notin S$. Combining this and $\mathcal{L}' \models c[\vec{t}/\vec{x}]$, we get $\mathcal{L}' \models (c \wedge \vec{x} \notin S)[\vec{t}/\vec{x}]$. Since $\mathcal{L}' \models \varphi'[\vec{t}/\vec{x}]$, we derive from the definition of $\models$ that $\mathcal{L}' \models \exists\vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')$. This immediately yields $\mathcal{L}' \models \psi$.

Conversely, suppose that $\mathcal{L}' \models \psi$. We show that $\mathcal{L}' \models \varphi$. $\mathcal{L}' \models \psi$ implies that either $\mathcal{L}' \models \psi_i$ for some i or $\mathcal{L}' \models \exists \vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')$. We consider both subcases below.

Subcase. $\mathcal{L} \models \psi_i$. Since $\mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}]) = \psi_i$, by the i.h., $\mathcal{L}' \models \varphi'[\vec{t_i}/\vec{x}]$. Further, observe that because $\sigma_i \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$ and $\mathtt{dom}(\sigma_i) \supseteq \vec{x} \supseteq \mathtt{fv}(c)$ (the latter because $\exists \vec{x}.(c \supset \varphi')$ must be closed), Theorem B.3 yields $\mathcal{L} \models c\sigma_i$ and, hence, $\mathcal{L} \models c[\vec{t_i}/\vec{x}]$. By Lemma B.1, $\mathcal{L}' \models c[\vec{t_i}/\vec{x}]$. Since we have already derived $\mathcal{L}' \models \varphi'[\vec{t_i}/\vec{x}]$, the definition of $\models$ yields that $\mathcal{L}' \models \exists \vec{x}.(c \wedge \varphi')$, as required.

Subcase. $\mathcal{L}' \models \exists \vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')$. Thus there must be a $\vec{t}$ such that $\mathcal{L}' \models c[\vec{t}/\vec{x}]$, $\vec{t} \notin S$, and $\mathcal{L}' \models \varphi'[\vec{t}/\vec{x}]$. The first and third facts in the last sentence imply that $\mathcal{L}' \models \exists \vec{x}.(c \wedge \varphi')$, as required. □

Theorem B.6 (Totality of $\widehat{\mathtt{sat}}$; Theorem 4.7)**.** *If $\chi_I \vdash c : \chi_O$, then for all structures $\mathcal{L}$ and all substitutions σ with $\mathtt{dom}(\sigma) \supseteq \chi_I$, $\widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$ is defined and, further, for each substitution $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_O$.*

Proof. By induction on the given derivation of $\chi_I \vdash c : \chi_O$ and case analysis of its last rule.

Case. $$\frac{\forall k \in I(p_O).\ \mathtt{fv}(t_k) \subseteq \chi_I \qquad \chi_O = \chi_I \cup (\bigcup_{j \in O(p_O)} \mathtt{fv}(t_j))}{\chi_I \vdash p_O(t_1, \ldots, t_n) : \chi_O}$$

We are given σ such that $\mathtt{dom}(\sigma) \supseteq \chi_I$. From this and the first premise it follows that $\forall k \in I(p_O).\ \mathtt{ground}(t_k\sigma)$. Thus, by definition, $\mathtt{sat}(\mathcal{L}, p_O(t_1, \ldots, t_n)\sigma)$ is defined. Consequently, $\widehat{\mathtt{sat}}(\mathcal{L}, p_O(t_1, \ldots, t_n)\sigma)$, which equals $\mathtt{sat}(\mathcal{L}, p_O(t_1, \ldots, t_n)\sigma)$ is also defined. Pick any $\sigma' \in \mathtt{sat}(\mathcal{L}, p_O(t_1, \ldots, t_n)\sigma)$. By definition of $\mathtt{sat}$, $\mathtt{dom}(\sigma') \supseteq \bigcup_{j \in O(p_O)} \mathtt{fv}(t_j)$. Consequently, $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_I \cup (\bigcup_{j \in O(p_O)} \mathtt{fv}(t_j)) \supseteq \chi_O$, where the last relation follows from the second premise.

Case. $$\frac{}{\chi_I \vdash \top : \chi_I}$$

Suppose $\mathtt{dom}(\sigma) \supseteq \chi_I$. Note that $\widehat{\mathtt{sat}}(\mathcal{L}, \top\sigma) = \widehat{\mathtt{sat}}(\mathcal{L}, \top) = \{\bullet\}$ is always defined. If $\sigma' \in \{\bullet\}$, then $\sigma' = \bullet$. Clearly, $\chi_I \cup \mathtt{dom}(\sigma') = \chi_I \cup \mathtt{dom}(\bullet) = \chi_I = \chi_O$.

Case. $$\frac{}{\chi_I \vdash \bot : \chi_I}$$

Suppose $\mathtt{dom}(\sigma) \supseteq \chi_I$. Note that $\widehat{\mathtt{sat}}(\mathcal{L}, \bot\sigma) = \widehat{\mathtt{sat}}(\mathcal{L}, \bot) = \{\}$ is always defined. Because there cannot be a $\sigma' \in \{\}$, the rest of the proof holds vacuously in this case.

Case. $$\frac{\chi_I \vdash c_1 : \chi \qquad \chi \vdash c_2 : \chi_O}{\chi_I \vdash c_1 \wedge c_2 : \chi_O}$$

Suppose $\mathtt{dom}(\sigma) \supseteq \chi_I$. By i.h. on the first premise, $\widehat{\mathtt{sat}}(\mathcal{L}, c_1\sigma)$ is defined. Let $\widehat{\mathtt{sat}}(\mathcal{L}, c_1\sigma) = \{\sigma_1, \ldots, \sigma_n\}$. Also by the i.h., $\chi_I \cup \mathtt{dom}(\sigma_i) \supseteq \chi$. Call this fact (A). Since $\mathtt{dom}(\sigma) \supseteq \chi_I$, fact (A) implies $\mathtt{dom}(\sigma + \sigma_i) \supseteq \chi$. Using the latter, by i.h. on the second premise and each of $\{\sigma + \sigma_1, \ldots, \sigma + \sigma_n\}$, we obtain that each of $\widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma\sigma_i)$ are also defined for each i and $\forall \sigma_i' \in \widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma\sigma_i)$, $\chi \cup \mathtt{dom}(\sigma_i') \supseteq \chi_O$. Call the last fact (B). We immediately have that $\widehat{\mathtt{sat}}(\mathcal{L}, (c_1 \wedge c_2)\sigma) = \bigcup_{\sigma_1 \in \widehat{\mathtt{sat}}(\mathcal{L}, c_1\sigma)} \widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma\sigma_1)$ is also defined.

Pick any $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, (c_1 \wedge c_2)\sigma)$. Then for some i and some $\sigma_i' \in \widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma\sigma_i)$, we have $\sigma' = \sigma_i + \sigma_i'$. We want to show that $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_O$. Or, equivalently, $\chi_I \cup \mathtt{dom}(\sigma_i + \sigma_i') \supseteq \chi_O$.

However, $\chi_I \cup \mathtt{dom}(\sigma_i + \sigma_i') = \chi_I \cup \mathtt{dom}(\sigma_i) \cup \mathtt{dom}(\sigma_i') \supseteq \chi \cup \mathtt{dom}(\sigma_i') \supseteq \chi_O$, where the last two relations follow from facts (A) and (B), respectively.

Case. $\dfrac{\chi_I \vdash c_1 : \chi_1 \qquad \chi_I \vdash c_2 : \chi_2}{\chi_I \vdash c_1 \vee c_2 : \chi_1 \cap \chi_2}$

Suppose $\mathtt{dom}(\sigma) \supseteq \chi_I$. By i.h. on the first premise, $\widehat{\mathtt{sat}}(\mathcal{L}, c_1\sigma)$ is defined and $\forall \sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c_1\sigma)$, $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_1$. Call this fact (A). Similarly, by i.h. on the second premise, $\widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma)$ is defined and $\forall \sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma)$, $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_2$. Call this fact (B). By definition of $\widehat{\mathtt{sat}}$, $\widehat{\mathtt{sat}}(\mathcal{L}, (c_1 \vee c_2)\sigma) = \widehat{\mathtt{sat}}(\mathcal{L}, c_1\sigma) \cup \widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma)$ is defined.

Pick any $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c_1\sigma) \cup \widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma)$. We want to show $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_O$. Either $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c_1\sigma)$ or $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c_2\sigma)$. Consider the former case (the other case is similar). We have $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_1 \supseteq \chi_1 \cap \chi_2 = \chi_O$, where the first relation follows from fact (A).

Case. $\dfrac{\chi_I \vdash c : \chi_O'}{\chi_I \vdash \exists x.c : \chi_O' \backslash \{x\}}$

Suppose $\mathtt{dom}(\sigma) \supseteq \chi_I$. By i.h. on the premise, $\widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$ is defined and $\forall \sigma'' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\chi_I \cup \mathtt{dom}(\sigma'') \supseteq \chi_O'$. Call the latter fact (A). By definition of $\widehat{\mathtt{sat}}$, $\widehat{\mathtt{sat}}(\mathcal{L}, \exists x.c) = \widehat{\mathtt{sat}}(\mathcal{L}, c)\backslash\{x\}$ is defined.

Pick any $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c)\backslash\{x\}$. We want to prove that $\chi_I \cup \mathtt{dom}(\sigma') \supseteq \chi_O'\backslash\{x\}$. However, $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c)\backslash\{x\}$ implies (by definition) that there is a $\sigma'' \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$ such that $\sigma' = \sigma''\backslash\{x\}$. Thus, $\chi_I \cup \mathtt{dom}(\sigma') = \chi_I \cup (\mathtt{dom}(\sigma'')\backslash\{x\}) \supseteq \chi_O'\backslash\{x\}$. The last inclusion follows from fact (A). □

Lemma B.7. *If $\chi_I \vdash c : \chi_O$, then $\chi_O \subseteq \chi_I \cup \mathtt{fv}(c)$.*

Proof. By a straightforward induction on the given derivation of $\chi_I \vdash c : \chi_O$. □

Lemma B.8 (Mode substitution)**.** *The following hold:*

1. *If $\chi_I \vdash c : \chi_O$, then $\chi_I\backslash\mathtt{dom}(\sigma) \vdash c\sigma : \chi_O\backslash\mathtt{dom}(\sigma)$.*
2. *If $\chi \vdash \varphi$, then $\chi\backslash\mathtt{dom}(\sigma) \vdash \varphi\sigma$.*

Proof. By induction on the given derivations of $\chi_I \vdash c : \chi_O$ and $\chi \vdash \varphi$. □

Lemma B.9 (Mode weakening)**.** *The following hold:*

1. *If $\chi_I \vdash c : \chi_O$ and $\chi_I' \supseteq \chi_I$, then there is a $\chi_O' \supseteq \chi_O$ such that $\chi_I' \vdash c : \chi_O'$.*
2. *If $\chi \vdash \varphi$ and $\chi' \supseteq \chi$, then $\chi' \vdash \varphi$.*

Proof. By induction on the given derivations of $\chi_I \vdash c : \chi_O$ and $\chi \vdash \varphi$. □

Theorem B.10 (Totality of reduce; Theorem 4.8)**.** *If $\vdash \varphi$ then there is a ψ such that $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$ and $\vdash \psi$.*

Proof. We prove a more general result: If $\chi \vdash \varphi$ and $\mathtt{dom}(\sigma) \supseteq \chi$, then there is a ψ such that $\mathsf{reduce}(\mathcal{L}, \varphi\sigma) = \psi\sigma$ and $\chi \vdash \psi$. The statement of the theorem follows by choosing $\chi = \{\}$ and $\sigma = \bullet$ in this result. We proceed by induction on the assumed derivation of $\chi \vdash \varphi$, and case analysis of its last rule.

Case. $\dfrac{\forall k.\ \mathtt{fv}(t_k) \subseteq \chi}{\chi \vdash p(t_1, \dots, t_k)}$

Here, $\varphi = p(t_1, \dots, t_k)$. Suppose $\mathtt{dom}(\sigma) \supseteq \chi$. Due to the premise, $p(t_1, \dots, t_n)\sigma$ is ground. Hence, $\widehat{\mathtt{sat}}(\mathcal{L}, p(t_1, \dots, t_n)\sigma)$ is defined. Depending on whether it is $\mathtt{tt}$, $\mathtt{ff}$, or $\mathtt{uu}$, $\mathsf{reduce}(\mathcal{L}, p(t_1, \dots, t_n)\sigma)$ is $\top$, $\bot$ or $p(t_1, \dots, t_n)\sigma$ respectively. Accordingly, we choose $\psi = \top$, $\psi = \bot$ or $\psi = p(t_1, \dots, t_n)$. In each case, $\chi \vdash \psi$.

Case. $\dfrac{}{\chi \vdash \top}$

Here, $\varphi = \top$. Suppose $\mathtt{dom}(\sigma) \supseteq \chi$. Clearly, we can choose $\psi = \top$ because $\mathsf{reduce}(\mathcal{L}, \top\sigma) = \top = \psi\sigma$ and $\chi \vdash \top$, i.e., $\chi \vdash \psi$.

Case. $\dfrac{}{\chi \vdash \bot}$

Here, $\varphi = \bot$. Suppose $\mathtt{dom}(\sigma) \supseteq \chi$. Clearly, we can choose $\psi = \bot$ because $\mathsf{reduce}(\mathcal{L}, \bot\sigma) = \bot = \psi\sigma$ and $\chi \vdash \bot$, i.e., $\chi \vdash \psi$.

Case. $\dfrac{\chi \vdash \varphi_1 \qquad \chi \vdash \varphi_2}{\chi \vdash \varphi_1 \wedge \varphi_2}$

Here, $\varphi = \varphi_1 \wedge \varphi_2$. Suppose $\mathtt{dom}(\sigma) \supseteq \chi$. By i.h. on the first premise, there is a ψ_1 such that $\mathsf{reduce}(\mathcal{L}, \varphi_1\sigma) = \psi_1\sigma$ and $\chi \vdash \psi_1$. Similarly, by i.h. on the second premise, there is a ψ_2 such that $\mathsf{reduce}(\mathcal{L}, \varphi_2\sigma) = \psi_2\sigma$ and $\chi \vdash \psi_2$. By definition of reduce, $\mathsf{reduce}(\mathcal{L}, \varphi\sigma) = \mathsf{reduce}(\mathcal{L}, (\varphi_1 \wedge \varphi_2)\sigma) = \mathsf{reduce}(\mathcal{L}, \varphi_1\sigma) \wedge \mathsf{reduce}(\mathcal{L}, \varphi_2\sigma) = \psi_1\sigma \wedge \psi_2\sigma$. Further, $\chi \vdash \psi_1 \wedge \psi_2$ follows from $\chi \vdash \psi_1$ and $\chi \vdash \psi_2$. So we can choose $\psi = \psi_1 \wedge \psi_2$.

Case. $\dfrac{\chi \vdash \varphi_1 \qquad \chi \vdash \varphi_2}{\chi \vdash \varphi_1 \vee \varphi_2}$

Here, $\varphi = \varphi_1 \vee \varphi_2$. Suppose $\mathtt{dom}(\sigma) \supseteq \chi$. By i.h. on the first premise, there is a ψ_1 such that $\mathsf{reduce}(\mathcal{L}, \varphi_1\sigma) = \psi_1\sigma$ and $\chi \vdash \psi_1$. Similarly, by i.h. on the second premise, there is a ψ_2 such that $\mathsf{reduce}(\mathcal{L}, \varphi_2\sigma) = \psi_2\sigma$ and $\chi \vdash \psi_2$. By definition of reduce, $\mathsf{reduce}(\mathcal{L}, \varphi\sigma) = \mathsf{reduce}(\mathcal{L}, (\varphi_1 \vee \varphi_2)\sigma) = \mathsf{reduce}(\mathcal{L}, \varphi_1\sigma) \vee \mathsf{reduce}(\mathcal{L}, \varphi_2\sigma) = \psi_1\sigma \vee \psi_2\sigma$. Further, $\chi \vdash \psi_1 \vee \psi_2$ follows from $\chi \vdash \psi_1$ and $\chi \vdash \psi_2$. So we can choose $\psi = \psi_1 \vee \psi_2$.

Case. $\dfrac{\chi \vdash c : \chi_O \qquad \vec{x} \subseteq \chi_O \qquad \mathtt{fv}(c) \subseteq \chi \cup \vec{x} \qquad \chi_O \vdash \varphi'}{\chi \vdash \forall \vec{x}.(c \supset \varphi')}$

Here, $\varphi = \forall \vec{x}.(c \supset \varphi')$. Suppose $\mathtt{dom}(\sigma) \supseteq \chi$. By Theorem B.6 on the first premise, there is a set $\{\sigma_1, \dots, \sigma_n\} = \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$ such that for each σ_i, $\chi \cup \mathtt{dom}(\sigma_i) \supseteq \chi_O$. Call the latter fact (A). From the second premise and fact (A) we also derive that $\chi \cup \mathtt{dom}(\sigma_i) \supseteq \vec{x}$. Since $\vec{x}$ must be chosen fresh in the premise, this also implies that $\mathtt{dom}(\sigma_i) \supseteq \vec{x}$. Consequently, $\sigma_i(\vec{x})$ is defined. Let $\sigma_i(\vec{x}) = \vec{t_i}$ and let $S = \{\vec{t_1}, \dots, \vec{t_n}\}$. Further, note that by Lemma B.7 on the first premise, $\chi_O \subseteq \chi \cup \mathtt{fv}(c)$. Hence, from the third premise we obtain $\chi_O \subseteq \chi \cup \chi \cup \vec{x} = \chi \cup \vec{x}$. So, $\mathtt{dom}(\sigma) \cup \vec{x} \supseteq \chi \cup \vec{x} \supseteq \chi_O$. Call this fact (B). From the i.h. applied to the last premise and fact (B) we get the existence of ψ_i such that $\mathsf{reduce}(\mathcal{L}, \varphi'\sigma[\vec{t_i}/\vec{x}]) = \psi_i\sigma[\vec{t_i}/\vec{x}]$ and $\chi_O \vdash \psi_i$. Call this fact (C).

By definition of reduce, we obtain $\mathsf{reduce}(\mathcal{L}, \varphi\sigma) = \psi_1\sigma[\vec{t_1}/\vec{x}] \wedge \dots \wedge \psi_n\sigma[\vec{t_n}/\vec{x}] \wedge \psi'\sigma$, where $\psi' = \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$. Choose $\psi = \psi_1[\vec{t_1}/\vec{x}] \wedge \dots \wedge \psi_n[\vec{t_n}/\vec{x}] \wedge \psi'$. It only remains to show that $\chi \vdash \psi$. This is equivalent to showing that $\chi \vdash \psi_i[\vec{t_i}/\vec{x}]$ and $\chi \vdash \psi'$. The latter, which is equal to

$\chi \vdash \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$, follows from the four premises of the rule above. It remains to show that $\chi \vdash \psi_i[\vec{t_i}/\vec{x}]$. Applying Lemma B.8 to fact(C), we derive that $\chi_O \backslash \vec{x} \vdash \psi_i[\vec{t_i}/\vec{x}]$. Since we already derived that $\chi_O \subseteq \chi \cup \vec{x}$, we also have $\chi_O \backslash \vec{x} \subseteq \chi$. Hence, by Lemma B.9, we get $\chi \vdash \psi_i[\vec{t_i}/\vec{x}]$, as required.

Case. $\dfrac{\chi \vdash c : \chi_O \qquad \vec{x} \subseteq \chi_O \qquad \mathtt{fv}(c) \subseteq \chi \cup \vec{x} \qquad \chi_O \vdash \varphi'}{\chi \vdash \exists \vec{x}.(c \wedge \varphi')}$

Here, $\varphi = \exists \vec{x}.(c \wedge \varphi')$. Suppose $\mathtt{dom}(\sigma) \supseteq \chi$. By Theorem B.6 on the first premise, there is a set $\{\sigma_1, \ldots, \sigma_n\} = \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$ such that for each σ_i, $\chi \cup \mathtt{dom}(\sigma_i) \supseteq \chi_O$. Call the latter fact (A). From the second premise and fact (A) we also derive that $\chi \cup \mathtt{dom}(\sigma_i) \supseteq \vec{x}$. Since $\vec{x}$ must be chosen fresh in the premise, this also implies that $\mathtt{dom}(\sigma_i) \supseteq \vec{x}$. Consequently, $\sigma_i(\vec{x})$ is defined. Let $\sigma_i(\vec{x}) = \vec{t_i}$ and let $S = \{\vec{t_1}, \ldots, \vec{t_n}\}$. Further, note that by Lemma B.7 on the first premise, $\chi_O \subseteq \chi \cup \mathtt{fv}(c)$. Hence, from the third premise we obtain $\chi_O \subseteq \chi \cup \chi \cup \vec{x} = \chi \cup \vec{x}$. So, $\mathtt{dom}(\sigma) \cup \vec{x} \supseteq \chi \cup \vec{x} \supseteq \chi_O$. Call this fact (B). From the i.h. applied to the last premise and fact (B) we get the existence of ψ_i such that $\mathsf{reduce}(\mathcal{L}, \varphi'\sigma[\vec{t_i}/\vec{x}]) = \psi_i\sigma[\vec{t_i}/\vec{x}]$ and $\chi_O \vdash \psi_i$. Call this fact (C).

By definition of reduce, we obtain $\mathsf{reduce}(\mathcal{L}, \varphi\sigma) = \psi_1\sigma[\vec{t_1}/\vec{x}] \vee \ldots \vee \psi_n\sigma[\vec{t_n}/\vec{x}] \vee \psi'\sigma$, where $\psi' = \exists \vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')$. Choose $\psi = \psi_1[\vec{t_1}/\vec{x}] \vee \ldots \vee \psi_n[\vec{t_n}/\vec{x}] \vee \psi'$. It only remains to show that $\chi \vdash \psi$. This is equivalent to showing that $\chi \vdash \psi_i[\vec{t_i}/\vec{x}]$ and $\chi \vdash \psi'$. The latter, which is equal to $\chi \vdash \exists \vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')$, follows from the four premises of the rule above. It remains to show that $\chi \vdash \psi_i[\vec{t_i}/\vec{x}]$. Applying Lemma B.8 to fact(C), we derive that $\chi_O \backslash \vec{x} \vdash \psi_i[\vec{t_i}/\vec{x}]$. Since we already derived that $\chi_O \subseteq \chi \cup \vec{x}$, we also have $\chi_O \backslash \vec{x} \subseteq \chi$. Hence, by Lemma B.9, we get $\chi \vdash \psi_i[\vec{t_i}/\vec{x}]$, as required. □

Lemma B.11 (Totality of $\mathtt{atoms}$)**.** *Suppose $\chi \vdash \varphi$ and $\mathtt{dom}(\sigma) \supseteq \chi$. Then, $\mathtt{atoms}(\mathcal{L}, \varphi\sigma)$ is defined and ground.*

Proof. By induction on the given derivation of $\chi \vdash \varphi$ and case analysis of its last rule.

Case. $\dfrac{\forall k.\ \mathtt{fv}(t_k) \subseteq \chi}{\chi \vdash p(t_1, \ldots, t_k)}$

Here $\varphi = p(t_1, \ldots, t_k)$. From the premise and given condition $\mathtt{dom}(\sigma) \supseteq \chi$, we know that $p(t_1, \ldots, t_k)\sigma$ is ground. Clearly, then $\mathtt{atoms}(\mathcal{L}, p(t_1, \ldots, t_k)\sigma) = \{p(t_1, \ldots, t_k)\sigma\}$ is defined and ground.

Case. $\dfrac{}{\chi \vdash \top}$

Here $\varphi = \top$. So $\mathtt{atoms}(\mathcal{L}, \varphi\sigma) = \mathtt{atoms}(\mathcal{L}, \top) = \{\}$ is defined and ground.

Case. $\dfrac{}{\chi \vdash \bot}$

Here $\varphi = \bot$. So $\mathtt{atoms}(\mathcal{L}, \varphi\sigma) = \mathtt{atoms}(\mathcal{L}, \bot) = \{\}$ is defined and ground.

Case. $\dfrac{\chi \vdash \varphi_1 \qquad \chi \vdash \varphi_2}{\chi \vdash \varphi_1 \wedge \varphi_2}$

Here $\varphi = \varphi_1 \wedge \varphi_2$. By the i.h. applied to the premises, $\mathtt{atoms}(\mathcal{L}, \varphi_i\sigma)$ for $i = 1, 2$ is defined and ground. It follows that $\mathtt{atoms}(\mathcal{L}, \varphi\sigma) = \mathtt{atoms}(\mathcal{L}, \varphi_1\sigma \wedge \varphi_2\sigma) = \mathtt{atoms}(\mathcal{L}, \varphi_1\sigma) \cup \mathtt{atoms}(\mathcal{L}, \varphi_2\sigma)$ is also defined and ground.

Case. $\dfrac{\chi \vdash \varphi_1 \qquad \chi \vdash \varphi_2}{\chi \vdash \varphi_1 \vee \varphi_2}$

Here $\varphi = \varphi_1 \vee \varphi_2$. By the i.h. applied to the premises, $\mathtt{atoms}(\mathcal{L}, \varphi_i\sigma)$ for $i = 1, 2$ is defined and ground. It follows that $\mathtt{atoms}(\mathcal{L}, \varphi\sigma) = \mathtt{atoms}(\mathcal{L}, \varphi_1\sigma \vee \varphi_2\sigma) = \mathtt{atoms}(\mathcal{L}, \varphi_1\sigma) \cup \mathtt{atoms}(\mathcal{L}, \varphi_2\sigma)$ is also defined and ground.

Case. $\dfrac{\chi \vdash c : \chi_O \qquad \vec{x} \subseteq \chi_O \qquad \mathtt{fv}(c) \subseteq \chi \cup \vec{x} \qquad \chi_O \vdash \varphi'}{\chi \vdash \forall \vec{x}.(c \supset \varphi')}$

Here $\varphi = \forall \vec{x}.(c \supset \varphi')$. By Theorem B.6 on the first premise and the given condition $\mathtt{dom}(\sigma) \supseteq \chi$, $\widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$ is defined and for all $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\chi \cup \mathtt{dom}(\sigma') \supseteq \chi_O$. The latter implies that for all $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\mathtt{dom}(\sigma\sigma') \supseteq \chi_O$. By i.h. on the last premise, for each $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\mathtt{atoms}(\mathcal{L}, \varphi'\sigma\sigma')$ is defined and ground. Hence, by definition, $\mathtt{atoms}(\mathcal{L}, \varphi\sigma) = \bigcup_{\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)} \mathtt{atoms}(\mathcal{L}, \varphi'\sigma\sigma')$ is defined and ground.

Case. $\dfrac{\chi \vdash c : \chi_O \qquad \vec{x} \subseteq \chi_O \qquad \mathtt{fv}(c) \subseteq \chi \cup \vec{x} \qquad \chi_O \vdash \varphi'}{\chi \vdash \exists \vec{x}.(c \wedge \varphi')}$

Here $\varphi = \exists \vec{x}.(c \wedge \varphi')$. By Theorem B.6 on the first premise and the given condition $\mathtt{dom}(\sigma) \supseteq \chi$, $\widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$ is defined and for all $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\chi \cup \mathtt{dom}(\sigma') \supseteq \chi_O$. The latter implies that for all $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\mathtt{dom}(\sigma\sigma') \supseteq \chi_O$. By i.h. on the last premise, for each $\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)$, $\mathtt{atoms}(\mathcal{L}, \varphi'\sigma\sigma')$ is defined and ground. Hence, by definition, $\mathtt{atoms}(\mathcal{L}, \varphi\sigma) = \bigcup_{\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c\sigma)} \mathtt{atoms}(\mathcal{L}, \varphi'\sigma\sigma')$ is defined and ground. □

Theorem B.12 (Minimality; Theorem 4.3)**.** *Suppose* $\vdash \varphi$ *and* $\mathsf{reduce}(\mathcal{L}, \varphi) = \psi$. *Then* $\mathtt{atoms}(\mathcal{L}, \psi) \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap \{P \mid \rho_{\mathcal{L}}(P) = \mathtt{uu}\}$.

Proof. By Lemma B.11, $\mathtt{atoms}(\mathcal{L}, \varphi)$ is defined. Further, by Theorem B.10, $\vdash \psi$, so $\mathtt{atoms}(\mathcal{L}, \psi)$ is also defined. Hence, the statement of the theorem makes sense. We prove the relation $\mathtt{atoms}(\mathcal{L}, \psi) \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap \{P \mid \rho_{\mathcal{L}}(P) = \mathtt{uu}\}$ by induction on φ and case analysis of its form. Let $U = \{P \mid \rho_{\mathcal{L}}(P) = \mathtt{uu}\}$. We want to show that $\mathtt{atoms}(\mathcal{L}, \psi) \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap U$.

Case. $\varphi = P$ where P is either a subjective or an objective atom. We perform a sub-case analysis on $\rho_{\mathcal{L}}(P)$.

Subcase. $\rho_{\mathcal{L}}(P) = \mathtt{tt}$. Then, $\psi = \top$. So, trivially, $\mathtt{atoms}(\psi) = \{\} \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap U$.

Subcase. $\rho_{\mathcal{L}}(P) = \mathtt{ff}$. Then, $\psi = \bot$. So, trivially, $\mathtt{atoms}(\psi) = \{\} \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap U$.

Subcase. $\rho_{\mathcal{L}}(P) = \mathtt{uu}$. Then, $\psi = P$. Further, in this case, $\mathtt{atoms}(\mathcal{L}, \psi) = \{P\} = \mathtt{atoms}(\mathcal{L}, \varphi)$ and $P \in U$ (the latter by definition of U). Clearly, $\mathtt{atoms}(\mathcal{L}, \psi) \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap U$.

Case. $\varphi = \top$. Here, $\psi = \top$. So, trivially, $\mathtt{atoms}(\psi) = \{\} \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap U$.

Case. $\varphi = \bot$. Here, $\psi = \bot$. So, trivially, $\mathtt{atoms}(\psi) = \{\} \subseteq \mathtt{atoms}(\mathcal{L}, \varphi) \cap U$.

Case. $\varphi = \varphi_1 \wedge \varphi_2$. Then, $\psi = \mathsf{reduce}(\mathcal{L}, \varphi_1) \wedge \mathsf{reduce}(\mathcal{L}, \varphi_2)$. By inversion on the derivation of $\vdash \varphi$, we know that $\vdash \varphi_1$ and $\vdash \varphi_2$. Hence, by the i.h., for $i = 1, 2$, $\mathtt{atoms}(\mathcal{L}, \mathsf{reduce}(\mathcal{L}, \varphi_i)) \subseteq \mathtt{atoms}(\mathcal{L}, \varphi_i) \cap U$. Thus, we have, $\mathtt{atoms}(\mathcal{L}, \psi) = \mathtt{atoms}(\mathcal{L}, \mathsf{reduce}(\mathcal{L}, \varphi_1)) \cup \mathtt{atoms}(\mathcal{L}, \mathsf{reduce}(\mathcal{L}, \varphi_2)) \subseteq (\mathtt{atoms}(\mathcal{L}, \varphi_1) \cap U) \cup (\mathtt{atoms}(\mathcal{L}, \varphi_2) \cap U) = (\mathtt{atoms}(\mathcal{L}, \varphi_1) \cup \mathtt{atoms}(\mathcal{L}, \varphi_2)) \cap U = \mathtt{atoms}(\mathcal{L}, \varphi) \cap U$.

Case. $\varphi = \varphi_1 \vee \varphi_2$. Then, $\psi = \mathsf{reduce}(\mathcal{L}, \varphi_1) \vee \mathsf{reduce}(\mathcal{L}, \varphi_2)$. By inversion on the derivation of $\vdash \varphi$, we know that $\vdash \varphi_1$ and $\vdash \varphi_2$. Hence, by the i.h., for $i = 1, 2$, $\mathtt{atoms}(\mathcal{L}, \mathsf{reduce}(\mathcal{L}, \varphi_i)) \subseteq \mathtt{atoms}(\mathcal{L}, \varphi_i) \cap U$. Thus, we have, $\mathtt{atoms}(\mathcal{L}, \psi) = \mathtt{atoms}(\mathcal{L}, \mathsf{reduce}(\mathcal{L}, \varphi_1)) \cup \mathtt{atoms}(\mathcal{L}, \mathsf{reduce}(\mathcal{L}, \varphi_2)) \subseteq (\mathtt{atoms}(\mathcal{L}, \varphi_1) \cap U) \cup (\mathtt{atoms}(\mathcal{L}, \varphi_2) \cap U) = (\mathtt{atoms}(\mathcal{L}, \varphi_1) \cup \mathtt{atoms}(\mathcal{L}, \varphi_2)) \cap U = \mathtt{atoms}(\mathcal{L}, \varphi) \cap U$.

Case. $\varphi = \forall \vec{x}.(c \supset \varphi')$. Then,

$$
\begin{array}{rcl}
\psi = \mathsf{reduce}(\mathcal{L}, \varphi) & = & \text{let} \\
 & & \quad \{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c) \\
 & & \quad \{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^{n} \\
 & & \quad S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\} \\
 & & \quad \{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])\}_{i=1}^{n} \\
 & & \quad \psi' \leftarrow \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi') \\
 & & \text{return} \\
 & & \quad \psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'
\end{array}
$$

By inversion on the given derivation of $\vdash \varphi$, we know that there is a χ_O such that (1) $\{\} \vdash c : \chi_O$, (2) $\vec{x} \subseteq \chi_O$, (3) $\mathtt{fv}(c) \subseteq \vec{x}$, and (4) $\chi_O \vdash \varphi'$. By Lemma B.7 on (1), $\chi_O \subseteq \mathtt{fv}(c)$. From this, (2), and (3), it follows that $\vec{x} = \mathtt{fv}(c) = \chi_O$. Call this fact (A). Using Lemma B.8 on (4), we get $\chi_O \backslash \vec{x} \vdash \varphi'[\vec{t_i}/\vec{x}]$. This and fact (A) imply that $\vdash \varphi'[\vec{t_i}/\vec{x}]$. Call this fact (B). By the i.h. on fact (B) and $\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])$, we get that $\mathtt{atoms}(\mathcal{L}, \psi_i) \subseteq \mathtt{atoms}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}]) \cap U$. Call this fact (C).

Next, $\widehat{\mathtt{sat}}(\mathcal{L}, (c \wedge \vec{x} \notin S)) = \bigcup_{\sigma' \in \widehat{\mathtt{sat}}(\mathcal{L}, c)} (\sigma' + \widehat{\mathtt{sat}}(\mathcal{L}, \sigma'(\vec{x}) \notin S)) = \bigcup_{i=1}^{n} (\sigma_i + \widehat{\mathtt{sat}}(\mathcal{L}, \vec{t_i} \notin S)) = \bigcup_{i=1}^{n} (\sigma_i + \{\}) = \{\}$. Hence, by definition, $\mathtt{atoms}(\mathcal{L}, \psi') = \mathtt{atoms}(\mathcal{L}, \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')) = \{\}$. Call this fact (D).

Also, $\mathtt{atoms}(\mathcal{L}, \varphi) = \mathtt{atoms}(\mathcal{L}, \forall \vec{x}.(c \supset \varphi')) = \bigcup_{\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)} \mathtt{atoms}(\mathcal{L}, \varphi'\sigma) = \bigcup_{i=1}^{n} \mathtt{atoms}(\mathcal{L}, \varphi'\sigma_i) = \bigcup_{i=1}^{n} \mathtt{atoms}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])$ (the last equality follows from $\mathtt{fv}(\varphi') \subseteq \vec{x}$, which in turn follows from fact (B)). Call this fact (E).

Finally, we have,

$$
\begin{array}{rcll}
\mathtt{atoms}(\mathcal{L}, \psi) & = & \mathtt{atoms}(\mathcal{L}, \psi_1 \wedge \ldots \wedge \psi_n \wedge \psi') & \\
 & = & \mathtt{atoms}(\mathcal{L}, \psi') \cup (\bigcup_{i=1}^{n} \mathtt{atoms}(\mathcal{L}, \psi_i)) & \text{(Defn. of } \mathtt{atoms}\text{)} \\
 & = & \{\} \cup (\bigcup_{i=1}^{n} \mathtt{atoms}(\mathcal{L}, \psi_i)) & \text{(Fact (D))} \\
 & = & \bigcup_{i=1}^{n} \mathtt{atoms}(\mathcal{L}, \psi_i) & \\
 & \subseteq & \bigcup_{i=1}^{n} (\mathtt{atoms}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}]) \cap U) & \text{(Fact (C))} \\
 & = & (\bigcup_{i=1}^{n} \mathtt{atoms}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])) \cap U & \\
 & = & \mathtt{atoms}(\mathcal{L}, \varphi) \cap U & \text{(Fact (E))}
\end{array}
$$

Case. $\varphi = \exists \vec{x}.(c \wedge \varphi')$. Then,

$$\begin{array}{rcl}
\psi = \mathsf{reduce}(\mathcal{L}, \varphi) & = & \text{let} \\
& & \quad \{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\texttt{sat}}(\mathcal{L}, c) \\
& & \quad \{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^{n} \\
& & \quad S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\} \\
& & \quad \{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])\}_{i=1}^{n} \\
& & \quad \psi' \leftarrow \exists \vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi') \\
& & \text{return} \\
& & \quad \psi_1 \vee \ldots \vee \psi_n \vee \psi'
\end{array}$$

By inversion on the given derivation of $\vdash \varphi$, we know that there is a χ_O such that (1) $\{\} \vdash c : \chi_O$, (2) $\vec{x} \subseteq \chi_O$, (3) $\texttt{fv}(c) \subseteq \vec{x}$, and (4) $\chi_O \vdash \varphi'$. By Lemma B.7 on (1), $\chi_O \subseteq \texttt{fv}(c)$. From this, (2), and (3), it follows that $\vec{x} = \texttt{fv}(c) = \chi_O$. Call this fact (A). Using Lemma B.8 on (4), we get $\chi_O \backslash \vec{x} \vdash \varphi'[\vec{t_i}/\vec{x}]$. This and fact (A) imply that $\vdash \varphi'[\vec{t_i}/\vec{x}]$. Call this fact (B). By the i.h. on fact (B) and $\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])$, we get that $\texttt{atoms}(\mathcal{L}, \psi_i) \subseteq \texttt{atoms}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}]) \cap U$. Call this fact (C).

Next, $\widehat{\texttt{sat}}(\mathcal{L}, (c \wedge \vec{x} \notin S)) = \bigcup_{\sigma' \in \widehat{\texttt{sat}}(\mathcal{L}, c)} (\sigma' + \widehat{\texttt{sat}}(\mathcal{L}, \sigma'(\vec{x}) \notin S)) = \bigcup_{i=1}^{n} (\sigma_i + \widehat{\texttt{sat}}(\mathcal{L}, \vec{t_i} \notin S)) = \bigcup_{i=1}^{n} (\sigma_i + \{\}) = \{\}$. Hence, by definition, $\texttt{atoms}(\mathcal{L}, \psi') = \texttt{atoms}(\mathcal{L}, \exists \vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')) = \{\}$. Call this fact (D).

Also, $\texttt{atoms}(\mathcal{L}, \varphi) = \texttt{atoms}(\mathcal{L}, \exists \vec{x}.(c \wedge \varphi')) = \bigcup_{\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c)} \texttt{atoms}(\mathcal{L}, \varphi'\sigma) = \bigcup_{i=1}^{n} \texttt{atoms}(\mathcal{L}, \varphi'\sigma_i) = \bigcup_{i=1}^{n} \texttt{atoms}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])$ (the last equality follows from $\texttt{fv}(\varphi') \subseteq \vec{x}$, which in turn follows from fact (B)). Call this fact (E).

Finally, we have,

$$\begin{array}{rcll}
\texttt{atoms}(\mathcal{L}, \psi) & = & \texttt{atoms}(\mathcal{L}, \psi_1 \vee \ldots \vee \psi_n \vee \psi') & \\
& = & \texttt{atoms}(\mathcal{L}, \psi') \cup (\bigcup_{i=1}^{n} \texttt{atoms}(\mathcal{L}, \psi_i)) & (\text{Defn. of } \texttt{atoms}) \\
& = & \{\} \cup (\bigcup_{i=1}^{n} \texttt{atoms}(\mathcal{L}, \psi_i)) & (\text{Fact (D)}) \\
& = & \bigcup_{i=1}^{n} \texttt{atoms}(\mathcal{L}, \psi_i) & \\
& \subseteq & \bigcup_{i=1}^{n} (\texttt{atoms}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}]) \cap U) & (\text{Fact (C)}) \\
& = & (\bigcup_{i=1}^{n} \texttt{atoms}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])) \cap U & \\
& = & \texttt{atoms}(\mathcal{L}, \varphi) \cap U & (\text{Fact (E)})
\end{array}$$

□

C Proofs from Section 5

This appendix contains proofs of theorems presented in Section 5.

Lemma C.1. *Suppose ψ does not contain any quantifiers or objective atoms. Then, $\psi \rightarrow^* \psi'$ such that (1) ψ' is either $\top$, or $\bot$, or contains only subjective atoms and the connectives $\wedge$, $\vee$, and (2) For all structures $\mathcal{L}$, $\mathcal{L} \models \psi$ iff $\mathcal{L} \models \psi'$ and $\mathcal{L} \models \overline{\psi}$ iff $\mathcal{L} \models \overline{\psi'}$.*

Proof. By induction on ψ. If ψ is either $\top$, $\bot$, or P_S, we can choose $\psi' = \psi$.

If $\psi = \psi_1 \wedge \psi_2$, then we inductively rewrite both ψ_1 and ψ_2 to ψ_1' and ψ_2', respectively. Thus, $\psi_1 \wedge \psi_2 \rightarrow^* \psi_1' \wedge \psi_2'$. If either ψ_1' or ψ_2' equals $\bot$, then $\psi_1' \wedge \psi_2' \rightarrow \bot$ and we choose $\psi' = \bot$. If $\psi_1' = \top$, then $\psi_1' \wedge \psi_2' \rightarrow \psi_2'$, so we can choose $\psi' = \psi_2'$. Similarly, if $\psi_2' = \top$, then $\psi_1' \wedge \psi_2' \rightarrow \psi_1'$, so we can choose $\psi' = \psi'_1$. Finally, if both ψ'_1 and ψ'_2 contain only subjective atoms and connectives $\wedge$, $\vee$, then we choose $\psi' = \psi'_1 \wedge \psi'_2$.

The case of $\psi = \psi_1 \vee \psi_2$ is similarly handles. No other cases apply. □

Lemma C.2. *If $\mathcal{L}$ is objectively-complete, then for all restrictions c, either $\mathcal{L} \models c$ or $\mathcal{L} \models \overline{c}$.*

Proof. By induction on c. □

Lemma C.3. *If $\mathcal{L}$ is objectively-complete and $\mathcal{L}' \geq \mathcal{L}$, then for all restrictions c, $\mathcal{L}' \models c$ iff $\mathcal{L} \models c$.*

Proof. Suppose $\mathcal{L}' \geq \mathcal{L}$. Observe that because $\mathcal{L}$ is objectively-complete, $\mathcal{L}'$ and $\mathcal{L}$ agree on valuation of objective atoms, which are the only atoms in c. The result now follows by a straightforward induction on c. □

Theorem C.4 (Theorem 5.2)**.** *Suppose $\mathcal{L}$ is objectively-complete, $\vdash \varphi$ and $\psi = \mathsf{reduce}(\mathcal{L}, \varphi)$. Then $\psi \to^* \psi'$, where (1) ψ' is either $\top$, or $\bot$, or contains only subjective atoms and the connectives $\wedge$, $\vee$, and (2) For all $\mathcal{L}' \geq \mathcal{L}$, $\mathcal{L}' \models \psi$ iff $\mathcal{L}' \models \psi'$ and $\mathcal{L}' \models \overline{\psi}$ iff $\mathcal{L}' \models \overline{\psi'}$.*

Proof. By induction on φ and case analysis of its form. Define $\mathtt{simp}(\psi')$ to mean statement (1) of the theorem, i.e., that ψ' is either $\top$, or $\bot$, or contains only subjective atoms and the connectives $\wedge$, $\vee$. Define $\mathtt{equiv}(\mathcal{L}, \psi, \psi')$ to mean statement (2) of the theorem, i.e., for all $\mathcal{L}' \geq \mathcal{L}$, $\mathcal{L}' \models \psi$ iff $\mathcal{L}' \models \psi'$ and $\mathcal{L}' \models \overline{\psi}$ iff $\mathcal{L}' \models \overline{\psi'}$.

Case. $\varphi = P_O$. In this case, $\rho_{\mathcal{L}}(P_O) \in \{\mathtt{tt}, \mathtt{ff}\}$ and, accordingly, $\psi = \top$ or $\psi = \bot$. So we can choose $\psi' = \psi$ to trivially satisfy both (1) and (2).

Case. $\varphi = P_S$. In this case $\psi = \top$ or $\psi = \bot$ or $\psi = P_S$. So we can choose $\psi' = \psi$ to trivially satisfy both (1) and (2).

Case. $\varphi = \top$. Then, $\psi = \top$. We choose $\psi' = \psi$ to trivially satisfy both (1) and (2).

Case. $\varphi = \bot$. Then, $\psi = \bot$. We choose $\psi' = \psi$ to trivially satisfy both (1) and (2).

Case. $\varphi = \varphi_1 \wedge \varphi_2$. Then, $\psi = \psi_1 \wedge \psi_2$, where $\psi_i = \mathsf{reduce}(\mathcal{L}, \varphi_i)$ for $i = 1, 2$. By inversion on the given derivation of $\vdash \varphi$, we deduce $\vdash \varphi_1$ and $\vdash \varphi_2$. Hence, from the i.h., $\psi_i \to^* \psi'_i$ where $\mathtt{simp}(\psi'_i)$ and $\mathtt{equiv}(\mathcal{L}, \psi_i, \psi'_i)$. The last fact implies that $\mathtt{equiv}(\mathcal{L}, \psi, \psi'_1 \wedge \psi'_2)$. Further, $\psi = \psi_1 \wedge \psi_2 \to^* \psi'_1 \wedge \psi'_2$. Using Lemma C.1, we obtain a ψ' such that $\psi'_1 \wedge \psi'_2 \to^* \psi'$, $\mathtt{simp}(\psi')$ and $\mathtt{equiv}(\mathcal{L}, \psi'_1 \wedge \psi'_2, \psi')$. The last fact and $\mathtt{equiv}(\mathcal{L}, \psi, \psi'_i \wedge \psi'_2)$ imply $\mathtt{equiv}(\mathcal{L}, \psi, \psi')$. So ψ' satisfies all our requirements.

Case. $\varphi = \varphi_1 \vee \varphi_2$. Then, $\psi = \psi_1 \vee \psi_2$, where $\psi_i = \mathsf{reduce}(\mathcal{L}, \varphi_i)$ for $i = 1, 2$. By inversion on the given derivation of $\vdash \varphi$, we deduce $\vdash \varphi_1$ and $\vdash \varphi_2$. Hence, from the i.h., $\psi_i \to^* \psi'_i$ where $\mathtt{simp}(\psi'_i)$ and $\mathtt{equiv}(\mathcal{L}, \psi_i, \psi'_i)$. The last fact implies that $\mathtt{equiv}(\mathcal{L}, \psi, \psi'_1 \vee \psi'_2)$. Further, $\psi = \psi_1 \vee \psi_2 \to^* \psi'_1 \vee \psi'_2$. Using Lemma C.1, we obtain a ψ' such that $\psi'_1 \vee \psi'_2 \to^* \psi'$, $\mathtt{simp}(\psi')$ and $\mathtt{equiv}(\mathcal{L}, \psi'_1 \vee \psi'_2, \psi')$. The last fact and $\mathtt{equiv}(\mathcal{L}, \psi, \psi'_i \vee \psi'_2)$ imply $\mathtt{equiv}(\mathcal{L}, \psi, \psi')$. So ψ' satisfies all our requirements.

Case. $\varphi = \forall\vec{x}.(c \supset \varphi')$. Then, ψ is calculated as follows:

```
$\psi = \mathsf{reduce}(\mathcal{L}, \varphi)$ = let
    $\{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c)$
    $\{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^{n}$
    $S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\}$
    $\{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])\}_{i=1}^{n}$
    $\psi'' \leftarrow \forall\vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$
  return
    $\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi''$
```

By inversion on the given derivation of $\vdash \varphi$, we know that there is a χ_O such that (1) $\{\} \vdash c : \chi_O$, (2) $\vec{x} \subseteq \chi_O$, (3) $\mathtt{fv}(c) \subseteq \vec{x}$, and (4) $\chi_O \vdash \varphi'$. By Lemma B.7 on (1), $\chi_O \subseteq \mathtt{fv}(c)$. From this, (2), and (3), it follows that $\vec{x} = \mathtt{fv}(c) = \chi_O$. Call this fact (A). Note also that by Theorem B.6, $\mathtt{dom}(\sigma_i) \supseteq \chi_O = \vec{x}$. Call this fact (B).

Next, we show that $\mathtt{equiv}(\mathcal{L}, \psi'', \top)$. Since for all $\mathcal{L}'$, $\mathcal{L}' \models \top$, it suffices to show that for all $\mathcal{L}' \geq \mathcal{L}$, $\mathcal{L}' \models \forall\vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$. By definition of $\models$, it suffices to prove that for all $\vec{t}$ and $\mathcal{L}' \geq \mathcal{L}$, $\mathcal{L}' \models \overline{c[\vec{t}/\vec{x}] \wedge \vec{t} \notin S}$, i.e., $\mathcal{L}' \models \overline{c[\vec{t}/\vec{x}]} \vee \vec{t} \in S$. If $\vec{t} = \vec{t_i}$ for some i, then $\mathcal{L}' \models \vec{t} \in S$ by definition of S, so we are done. Hence, we need only consider the case where $\vec{t} \notin S$. In this case we show that $\mathcal{L}' \models \overline{c[\vec{t}/\vec{x}]}$. By Lemma C.2, this is implied by $\mathcal{L}' \not\models c[\vec{t}/\vec{x}]$, so we show the latter. Suppose, for the sake of contradiction, that $\mathcal{L}' \models c[\vec{t}/\vec{x}]$. By Lemma C.3, $\mathcal{L} \models c[\vec{t}/\vec{x}]$. Hence, by Theorem B.3, there is a $\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$ such that $[\vec{x} \mapsto \vec{t}] \geq \sigma$. $\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$ forces $\sigma = \sigma_i$ for some i and, by fact (B), $\vec{t} = \vec{t_i}$. Hence, $\vec{t} = \vec{t_i} \in S$, a contradiction. Therefore, $\mathtt{equiv}(\mathcal{L}, \psi'', \top)$. Call this fact (C).

By Lemma B.8 on (4), we derive $\chi_O \backslash \vec{x} \vdash \varphi'[\vec{t}/\vec{x}]$. Using fact (A), we have $\vdash \varphi'[\vec{t}/\vec{x}]$. Applying the i.h. to this and $\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])$, we know that there is a ψ'_i such that $\psi_i \rightarrow^* \psi'_i$, $\mathtt{simp}(\psi'_i)$ and $\mathtt{equiv}(\mathcal{L}, \psi_i, \psi'_i)$. Call this fact (D).

Note that $\psi = \psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'' \rightarrow^* \psi'_1 \wedge \ldots \wedge \psi'_n \wedge \top$ (the second relation follows because $\psi'' \rightarrow \top$). Further, because $\mathtt{equiv}(\mathcal{L}, \psi_i, \psi'_i)$ (fact (D)) and $\mathtt{equiv}(\mathcal{L}, \psi'', \top)$ (fact (C)), it follows that $\mathtt{equiv}(\mathcal{L}, \psi, (\psi'_1 \wedge \ldots \wedge \psi'_n \wedge \top))$. Also, from fact (C), $\mathtt{simp}(\psi'_1 \wedge \ldots \wedge \psi'_n \wedge \top)$. The proof is complete by choosing the ψ' obtained by applying Lemma C.1 to $\psi'_1 \wedge \ldots \wedge \psi'_n \wedge \top$.

Case. $\varphi = \exists\vec{x}.(c \wedge \varphi')$. Then, ψ is calculated as follows:

```
$\psi = \mathsf{reduce}(\mathcal{L}, \varphi)$ = let
    $\{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c)$
    $\{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^{n}$
    $S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\}$
    $\{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])\}_{i=1}^{n}$
    $\psi'' \leftarrow \exists\vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')$
  return
    $\psi_1 \vee \ldots \vee \psi_n \vee \psi''$
```

By inversion on the given derivation of $\vdash \varphi$, we know that there is a χ_O such that (1) $\{\} \vdash c : \chi_O$, (2) $\vec{x} \subseteq \chi_O$, (3) $\mathtt{fv}(c) \subseteq \vec{x}$, and (4) $\chi_O \vdash \varphi'$. By Lemma B.7 on (1), $\chi_O \subseteq \mathtt{fv}(c)$. From this, (2), and (3), it follows that $\vec{x} = \mathtt{fv}(c) = \chi_O$. Call this fact (A). Note also that by Theorem B.6, $\mathtt{dom}(\sigma_i) \supseteq \chi_O = \vec{x}$. Call this fact (B).

Next, we show that $\texttt{equiv}(\mathcal{L}, \psi'', \bot)$. Since for all $\mathcal{L}'$, $\mathcal{L}' \models \overline{\bot} = \top$, it suffices to show that for all $\mathcal{L}' \geq \mathcal{L}$, $\mathcal{L}' \models \overline{\exists \vec{x}.((c \wedge \vec{x} \notin S) \wedge \varphi')}$, i.e., $\mathcal{L}' \models \forall \vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$. By definition of $\models$, it suffices to prove that for all $\vec{t}$ and $\mathcal{L}' \geq \mathcal{L}$, $\mathcal{L}' \models \overline{c[\vec{t}/\vec{x}] \wedge \vec{t} \notin S}$, i.e., $\mathcal{L}' \models \overline{c[\vec{t}/\vec{x}]} \vee \vec{t} \in S$. If $\vec{t} = \vec{t_i}$ for some i, then $\mathcal{L}' \models \vec{t} \in S$ by definition of S, so we are done. Hence, we need only consider the case where $\vec{t} \notin S$. In this case we show that $\mathcal{L}' \models \overline{c[\vec{t}/\vec{x}]}$. By Lemma C.2, this is implied by $\mathcal{L}' \not\models c[\vec{t}/\vec{x}]$, so we show the latter. Suppose, for the sake of contradiction, that $\mathcal{L}' \models c[\vec{t}/\vec{x}]$. By Lemma C.3, $\mathcal{L} \models c[\vec{t}/\vec{x}]$. Hence, by Theorem B.3, there is a $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c)$ such that $[\vec{x} \mapsto \vec{t}] \geq \sigma$. $\sigma \in \widehat{\texttt{sat}}(\mathcal{L}, c)$ forces $\sigma = \sigma_i$ for some i and, by fact (B), $\vec{t} = \vec{t_i}$. Hence, $\vec{t} = \vec{t_i} \in S$, a contradiction. Therefore, $\texttt{equiv}(\mathcal{L}, \psi'', \bot)$. Call this fact (C).

By Lemma B.8 on (4), we derive $\chi_O \backslash \vec{x} \vdash \varphi'[\vec{t}/\vec{x}]$. Using fact (A), we have $\vdash \varphi'[\vec{t}/\vec{x}]$. Applying the i.h. to this and $\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])$, we know that there is a ψ_i' such that $\psi_i \rightarrow^* \psi_i'$, $\texttt{simp}(\psi_i')$ and $\texttt{equiv}(\mathcal{L}, \psi_i, \psi_i')$. Call this fact (D).

Note that $\psi = \psi_1 \vee \ldots \vee \psi_n \vee \psi'' \rightarrow^* \psi_1' \vee \ldots \vee \psi_n' \vee \bot$ (the second relation follows because $\psi'' \rightarrow \bot$). Further, because $\texttt{equiv}(\mathcal{L}, \psi_i, \psi_i')$ (fact (D)) and $\texttt{equiv}(\mathcal{L}, \psi'', \bot)$ (fact (C)), it follows that $\texttt{equiv}(\mathcal{L}, \psi, (\psi_1' \vee \ldots \vee \psi_n' \vee \bot))$. Also, from fact (C), $\texttt{simp}(\psi_1' \vee \ldots \vee \psi_n' \vee \bot)$. The proof is complete by choosing the ψ' obtained by applying Lemma C.1 to $\psi_1' \vee \ldots \vee \psi_n' \vee \bot$. $\square$

Next, we turn to proofs of Theorems 5.4 and 5.5. Both theorems rely on a central lemma (Lemma C.11). In order to prove the lemma cleanly, we need a few definitions and some other lemmas. Note that in the rest of this Appendix we assume that there are no subjective predicates.

Definition C.5 (Protected restrictions)**.** Let T be a set of time points (possibly non-ground). We define a subclass "T-protected" of restrictions c of the sublogic inductively as follows:

1. $p_O(t_1, \ldots, t_n, \tau_0)$ is T-protected if $\tau_0 \in T$
2. $\vec{x} \notin S$ is T-protected
3. $\tau \neq \tau'$ is T-protected
4. $\texttt{in}(\tau, \tau', \tau_0)$ is T-protected if $\tau_0 \in T$
5. $\top$ is T-protected
6. $\bot$ is T-protected
7. $c_1 \wedge c_2$ is T-protected if both c_1 and c_2 are T-protected.
8. $c_1 \vee c_2$ is T-protected if both c_1 and c_2 are T-protected.
9. $\exists x.c$ is T-protected if c is T-protected.

Definition C.6 (Protected formulas)**.** Let T be a set of time points (possibly non-ground). We define a subclass "T-protected" of formulas φ of the sublogic inductively as follows:

1. $p_O(t_1, \ldots, t_n, \tau_0)$ is T-protected if $\tau_0 \in T$
2. $\top$ is T-protected
3. $\bot$ is T-protected

4. $\varphi_1 \wedge \varphi_2$ is T-protected if both φ_1 and φ_2 are T-protected

5. $\varphi_1 \vee \varphi_2$ is T-protected if both φ_1 and φ_2 are T-protected

6. $\forall \vec{x}.(c \supset \varphi)$ is T-protected if c is T-protected and φ is T-protected

7. $\forall \tau.((\mathtt{in}(\tau, \tau', \tau_0) \wedge c) \supset \varphi)$ is T-protected if c is T-protected, $\tau_0 \in T$, and φ is $(T \cup \{\tau\})$-protected

8. $\exists \vec{x}.(c \wedge \varphi)$ is T-protected if c is T-protected and φ is T-protected

9. $\exists \tau.((\mathtt{in}(\tau, \tau', \tau_0) \wedge c) \wedge \varphi)$ is T-protected if c is T-protected, $\tau_0 \in T$, and φ is $(T \cup \{\tau\})$-protected

Lemma C.7 (Excluded middle for protected formulas)**.** *Let T, τ_0 be ground. Suppose $\mathcal{L}$ is τ_0-complete and for all $\tau \in T$, $\tau \leq \tau_0$. Then, the following hold.*

1. *If c is ground and T-protected, then either $\mathcal{L} \models c$ or $\mathcal{L} \models \overline{c}$.*

2. *If φ is ground and T-protected, then either $\mathcal{L} \models \varphi$ or $\mathcal{L} \models \overline{\varphi}$.*

Proof. Both statements follow by an induction on the respective definitions of T-protected. We show some representative cases below.

Proof of (1).

Case. $c = p_O(t_1, \ldots, t_n, \tau)$ and $\tau \in T$. By definition of τ_0-complete and the fact $\tau \leq \tau_0$, we know that either $\rho_{\mathcal{L}}(p_O(t_1, \ldots, t_n, \tau)) = \mathtt{tt}$ or $\rho_{\mathcal{L}}(p_O(t_1, \ldots, t_n, \tau)) = \mathtt{ff}$. In the former case, $\mathcal{L} \models p_O(t_1, \ldots, t_n, \tau)$, while in the latter case, $\mathcal{L} \models \overline{p_O}(t_1, \ldots, t_n, \tau)$.

Case. $c = c_1 \wedge c_2$ and both c_1 and c_2 are T-protected. By the i.h., for each i, either $\mathcal{L} \models c_i$ or $\mathcal{L} \models \overline{c_i}$. If $\mathcal{L} \models c_1$ and $\mathcal{L} \models c_2$, then $\mathcal{L} \models c_1 \wedge c_2$, as required. If, on the other hand, for some i, $\mathcal{L} \models \overline{c_i}$, then $\mathcal{L} \models \overline{c_1} \vee \overline{c_2}$, i.e., $\mathcal{L} \models \overline{c}$.

Case. $c = \exists x.c$ and c is T-protected. By the i.h., for every t, either $\mathcal{L} \models c[t/x]$ or $\mathcal{L} \models \overline{c}[t/x]$. If there is a t such that $\mathcal{L} \models c[t/x]$, then also $\mathcal{L} \models \exists x.c$. If, on the other hand, for every t, $\mathcal{L} \models \overline{c}[t/x]$, then also, $\mathcal{L} \models \forall x.\overline{c}$, i.e., $\mathcal{L} \models \overline{\exists x.c}$.

Proof of (2).

Case. $\varphi = \forall \vec{x}.(c \supset \varphi')$ where c is T-protected and φ' is T-protected. If for any $\vec{t}$, $\mathcal{L} \models c[\vec{t}/\vec{x}]$ and $\mathcal{L} \models \overline{\varphi'}[\vec{t}/\vec{x}]$, then, by definition, $\mathcal{L} \models \exists \vec{x}.(c \wedge \overline{\varphi'})$, i.e., $\mathcal{L} \models \overline{\varphi}$ and we are done. Hence, we need only consider the case where for every $\vec{t}$, either $\mathcal{L} \not\models c[\vec{t}/\vec{x}]$ or $\mathcal{L} \not\models \overline{\varphi'}[\vec{t}/\vec{x}]$. However, by (1) and the i.h., we also deduce in this case that for every $\vec{t}$, either $\mathcal{L} \models \overline{c}[\vec{t}/\vec{x}]$ or $\mathcal{L} \models \varphi'[\vec{t}/\vec{x}]$. By definition of $\models$, $\mathcal{L} \models \varphi$ in this case.

Case. $\forall \tau.((\mathtt{in}(\tau, \tau', \tau_1) \wedge c) \supset \varphi')$ where c is T-protected, $\tau_1 \in T$, and φ' is $(T \cup \{\tau\})$-protected. We consider two exhaustive subcases:

Subcase. There is a ground τ'' such that $\mathcal{L} \models \mathtt{in}(\tau'', \tau', \tau_1)$, $\mathcal{L} \models c[\tau''/\tau]$ and $\mathcal{L} \models \overline{\varphi'}[\tau''/\tau]$. By definition of $\models$, $\mathcal{L} \models \exists\tau.((\mathtt{in}(\tau'', \tau', \tau_1) \wedge c) \wedge \overline{\varphi'})$, i.e., $\mathcal{L} \models \overline{\varphi}$.

Subcase. For every ground τ'', either $\mathcal{L} \not\models \mathtt{in}(\tau'', \tau', \tau_1)$, or $\mathcal{L} \not\models c[\tau''/\tau]$, or $\mathcal{L} \not\models \overline{\varphi'}[\tau''/\tau]$. In this case we show that $\mathcal{L} \models \varphi$. Following the definition of $\models$, pick any τ''. It suffices to prove that either $\mathcal{L} \models \overline{\mathtt{in}}(\tau'', \tau', \tau_1)$ or $\mathcal{L} \models \overline{c}[\tau''/\tau]$ or $\mathcal{L} \models \varphi'[\tau''/\tau]$. From the subcase assumption, $\mathcal{L} \not\models \mathtt{in}(\tau'', \tau', \tau_1)$, or $\mathcal{L} \not\models c[\tau''/\tau]$, or $\mathcal{L} \not\models \overline{\varphi'}[\tau''/\tau]$. If $\mathcal{L} \not\models \mathtt{in}(\tau'', \tau', \tau_1)$, then because $\mathtt{in}(\tau'', \tau', \tau_1)$ is T-protected (note that $\tau_1 \in T$), (1) implies that $\mathcal{L} \models \overline{\mathtt{in}}(\tau'', \tau', \tau_1)$. The case $\mathcal{L} \not\models c[\tau''/\tau]$ is similar. That leaves only the last case: $\mathcal{L} \not\models \overline{\varphi'}[\tau''/\tau]$. Since φ' is $(T \cup \{\tau\})$-protected, $\varphi'[\tau''/\tau]$ is $(T \cup \{\tau''\})$-protected. Further, because we already considered the case $\mathcal{L} \not\models \mathtt{in}(\tau'', \tau', \tau_1)$, we may assume here that $\mathcal{L} \models \mathtt{in}(\tau'', \tau', \tau_1)$, which implies $\tau'' \leq \tau_1 \leq \tau_0$. Thus, we can apply the i.h. to $\varphi'[\tau''/\tau]$ to deduce that either $\mathcal{L} \models \varphi'[\tau''/\tau]$ or $\mathcal{L} \models \overline{\varphi'}[\tau''/\tau]$. The latter is assumed to be false, so we must have $\mathcal{L} \models \varphi'[\tau''/\tau]$, as required. □

Lemma C.8 (Reduction of protected formulas)**.** *Let T, τ_0 be ground. Suppose φ is T-protected, $\vdash \varphi$, $\mathcal{L}$ is τ_0-complete, and for all $\tau \in T$, $\tau \leq \tau_0$. Then, $\mathsf{reduce}(\mathcal{L}, \varphi) \rightarrow^* \psi$, where $\psi = \top$ or $\psi = \bot$ and $\mathcal{L} \models \varphi$ iff $\mathcal{L} \models \psi$.*

Proof. By induction on the derivation of φ being T-protected. The proof is very similar to that of Theorem C.4 and we show here only some representative cases of the induction.

Case. $\varphi = p_O(t_1, \ldots, t_n, \tau)$ where $\tau \in T$. Because $\mathcal{L}$ is τ_0-complete and $\tau \leq \tau_0$, we know that $\rho_{\mathcal{L}}(p_O(t_1, \ldots, t_n, \tau)) \in \mathtt{tt}, \mathtt{ff}$. Accordingly, $\mathsf{reduce}(\mathcal{L}, \varphi) \in \{\top, \bot\}$, so we can choose $\psi = \mathsf{reduce}(\mathcal{L}, \varphi)$ to satisfy the theorem's requirements.

Case. $\varphi = \forall\vec{x}.(c \supset \varphi')$ where c and φ' are both T-protected. Then, $\mathsf{reduce}(\mathcal{L}, \varphi)$ is calculated as follows.

```
$\mathsf{reduce}(\mathcal{L}, \varphi)$ = let
    $\{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, c)$
    $\{\vec{t_i} \leftarrow \sigma_i(\vec{x})\}_{i=1}^n$
    $S \leftarrow \{\vec{t_1}, \ldots, \vec{t_n}\}$
    $\{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])\}_{i=1}^n$
    $\psi' \leftarrow \forall\vec{x}.((c \wedge \vec{x} \notin S) \supset \varphi')$
  return
    $\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'$
```

By inversion on the given derivation of $\vdash \varphi$, we know that there is a χ_O such that (1) $\{\} \vdash c : \chi_O$, (2) $\vec{x} \subseteq \chi_O$, (3) $\mathtt{fv}(c) \subseteq \vec{x}$, and (4) $\chi_O \vdash \varphi'$. By Lemma B.7 on (1), $\chi_O \subseteq \mathtt{fv}(c)$. From this, (2), and (3), it follows that $\vec{x} = \mathtt{fv}(c) = \chi_O$. Call this fact (A). Note also that by Theorem B.6, $\mathtt{dom}(\sigma_i) \supseteq \chi_O = \vec{x}$. Call this fact (B).

Next, we show that $\mathcal{L} \models \psi'$. Following the definition of $\models$, it suffices to prove that for all $\vec{t}$, $\mathcal{L} \models \overline{c[\vec{t}/\vec{x}] \wedge \vec{t} \notin S}$, i.e., either $\mathcal{L} \models \overline{c}[\vec{t}/\vec{x}]$ or $\vec{t} \in S$. Suppose $\vec{t} \notin S$. Then, we show that $\mathcal{L} \models \overline{c}[\vec{t}/\vec{x}]$. Because c is T-protected, Lemma C.7(1) applies, so the last fact is implied by $\mathcal{L} \not\models c[\vec{t}/\vec{x}]$. So we prove this instead. Suppose, for the sake of contradiction, that $\mathcal{L} \models c[\vec{t}/\vec{x}]$. Then, by Theorem B.3, there is a $\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$ such that $[\vec{x} \mapsto \vec{t}] \geq \sigma$. $\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, c)$ forces $\sigma = \sigma_i$ for some i and, by fact (B), $\vec{t} = \vec{t_i}$. Hence, $\vec{t} = \vec{t_i} \in S$, a contradiction. Hence, we must have $\mathcal{L} \models \psi'$. Call this fact (C).

By Lemma B.8 on (4), we derive $\chi_O \backslash \vec{x} \vdash \varphi'[\vec{t}/\vec{x}]$. Using fact (A), we have $\vdash \varphi'[\vec{t}/\vec{x}]$. We already know that φ' is T-protected and, hence, $\varphi'[\vec{t}/\vec{x}]$ is also T-protected. Applying the i.h. to the last two facts, and $\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\vec{t_i}/\vec{x}])$, we know that there is a $\psi_i' \in \{\top, \bot\}$ such that $\psi_i \rightarrow^* \psi_i'$ and $\mathcal{L} \models \varphi'[\vec{t_i}/\vec{x}]$ iff $\mathcal{L} \models \psi_i'$. Note that by Theorem B.5, this also implies $\mathcal{L} \models \psi_i$ iff $\mathcal{L} \models \psi_i'$. Call this fact (D). We consider two subcases:

Subcase. For every i, $\psi_i' = \top$. Clearly, we have $\mathsf{reduce}(\mathcal{L}, \varphi) = (\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi') \rightarrow^* \top$ (note: $\psi' \rightarrow \top$). We must show that $\mathcal{L} \models \varphi$. We have by fact (D) that $\mathcal{L} \models \psi_i$ for each i and by fact (C) that $\mathcal{L} \models \psi'$. Consequently, $\mathcal{L} \models (\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi')$ and, hence, by Theorem B.5, $\mathcal{L} \models \varphi$.

Subcase. There is a i such that $\psi_i' = \bot$. Clearly, we have $\mathsf{reduce}(\mathcal{L}, \varphi) = (\ldots \wedge \psi_i \wedge \ldots) \rightarrow^* \bot$. We must show that $\mathcal{L} \not\models \varphi$. Note that by fact (D), $\mathcal{L} \not\models \psi_i$. Consequently, by definition of $\models$, $\mathcal{L} \not\models \mathsf{reduce}(\mathcal{L}, \varphi)$ and, hence, by Theorem B.5, $\mathcal{L} \not\models \varphi$, as required.

Case. $\varphi = \forall x.((\mathtt{in}(x, \tau', \tau) \wedge c) \supset \varphi')$ where c is T-protected, $\tau \in T$, and φ' is $(T \cup \{x\})$-protected. Then, $\mathsf{reduce}(\mathcal{L}, \varphi)$ is calculated as follows.

$$
\begin{array}{rcl}
\mathsf{reduce}(\mathcal{L}, \varphi) & = & \text{let} \\
 & & \quad \{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, (\mathtt{in}(x, \tau', \tau) \wedge c)) \\
 & & \quad \{\tau_i \leftarrow \sigma_i(x)\}_{i=1}^n \\
 & & \quad S \leftarrow \{\tau_1, \ldots, \tau_n\} \\
 & & \quad \{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\tau_i/x])\}_{i=1}^n \\
 & & \quad \psi' \leftarrow \forall x.((\mathtt{in}(x, \tau', \tau) \wedge c \wedge x \notin S) \supset \varphi') \\
 & & \text{return} \\
 & & \quad \psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'
\end{array}
$$

By inversion on the given derivation of $\vdash \varphi$, we know that there is a χ_O such that (1) $\{\} \vdash \mathtt{in}(x, \tau', \tau) \wedge c : \chi_O$, (2) $\{x\} \subseteq \chi_O$, (3) $\mathtt{fv}(\mathtt{in}(x, \tau', \tau) \wedge c) \subseteq \{x\}$, and (4) $\chi_O \vdash \varphi'$. By Lemma B.7 on (1), $\chi_O \subseteq \mathtt{fv}(\mathtt{in}(x, \tau', \tau) \wedge c)$. From this, (2), and (3), it follows that $\{x\} = \mathtt{fv}(\mathtt{in}(x, \tau', \tau) \wedge c) = \chi_O$. Call this fact (A). Note also that by Theorem B.6, $\mathtt{dom}(\sigma_i) \supseteq \chi_O = \{x\}$. Call this fact (B).

Next, we show that $\mathcal{L} \models \psi'$. Following the definition of $\models$, it suffices to prove that for all t, $\mathcal{L} \models \overline{\mathtt{in}(t, \tau', \tau) \wedge c[t/x] \wedge t \notin S}$, i.e., either $\mathcal{L} \models \overline{\mathtt{in}(t, \tau', \tau) \wedge c[t/x]}$ or $t \in S$. Suppose $t \notin S$. Then, we show that $\mathcal{L} \models \overline{\mathtt{in}(t, \tau', \tau) \wedge c[t/x]}$. Because $\mathtt{in}(t, \tau', \tau) \wedge c[t/x]$ is T-protected, Lemma C.7(1) applies, so the last fact is implied by $\mathcal{L} \not\models \mathtt{in}(t, \tau', \tau) \wedge c[t/x]$. So we prove this instead. Suppose, for the sake of contradiction, that $\mathcal{L} \models \mathtt{in}(t, \tau', \tau) \wedge c[t/x]$. Then, by Theorem B.3, there is a $\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, \mathtt{in}(x, \tau', \tau) \wedge c)$ such that $[x \mapsto t] \geq \sigma$. $\sigma \in \widehat{\mathtt{sat}}(\mathcal{L}, \mathtt{in}(x, \tau', \tau) \wedge c)$ forces $\sigma = \sigma_i$ for some i and, by fact (B), $t = \tau_i$. Hence, $t = \tau_i \in S$, a contradiction. Hence, we must have $\mathcal{L} \models \psi'$. Call this fact (C).

By Lemma B.8 on (4), we derive $\chi_O \backslash \{x\} \vdash \varphi'[\tau_i/x]$. Using fact (A), we have $\vdash \varphi'[\tau_i/x]$. We already know that φ' is $(T \cup \{x\})$-protected and, hence, $\varphi'[\tau_i/x]$ is $(T \cup \{\tau_i\})$-protected. Note also that $\tau_i \leq \tau \leq \tau_0$. Applying the i.h. to the last three facts, and $\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, \varphi'[\tau_i/x])$, we know that there is a $\psi_i' \in \{\top, \bot\}$ such that $\psi_i \rightarrow^* \psi_i'$ and $\mathcal{L} \models \varphi'[\tau_i/x]$ iff $\mathcal{L} \models \psi_i'$. Note that by Theorem B.5, this also implies $\mathcal{L} \models \psi_i$ iff $\mathcal{L} \models \psi_i'$. Call this fact (D). We consider two subcases:

Subcase. For every i, $\psi_i' = \top$. Clearly, we have $\mathsf{reduce}(\mathcal{L}, \varphi) = (\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi') \rightarrow^* \top$ (note: $\psi' \rightarrow \top$). We must show that $\mathcal{L} \models \varphi$. We have by fact (D) that $\mathcal{L} \models \psi_i$ for each i and by fact (C) that $\mathcal{L} \models \psi'$. Consequently, $\mathcal{L} \models (\psi_1 \wedge \ldots \wedge \psi_n \wedge \psi')$ and, hence, by Theorem B.5, $\mathcal{L} \models \varphi$.

Subcase. There is a i such that $\psi'_i = \bot$. Clearly, we have $\mathsf{reduce}(\mathcal{L}, \varphi) = (\ldots \wedge \psi_i \wedge \ldots) \rightarrow^* \bot$. We must show that $\mathcal{L} \not\models \varphi$. Note that by fact (D), $\mathcal{L} \not\models \psi_i$. Consequently, by definition of $\models$, $\mathcal{L} \not\models \mathsf{reduce}(\mathcal{L}, \varphi)$ and, hence, by Theorem B.5, $\mathcal{L} \not\models \varphi$, as required. □

Lemma C.9 (Duality of protection). *φ is T-protected iff $\overline{\varphi}$ is T-protected.*

Proof. By a straightforward induction on φ. □

Lemma C.10 (Past translation). *The following hold:*

1. *If c is a restriction in the temporal logic, then for any $\tau \in T$, $(c)^\tau$ is T-protected.*
2. *If α_p is a temporal logic formula without future operators, then for any $\tau \in T$, $(\alpha_p)^\tau$ is T-protected.*

Proof. (1) follows by a straightforward induction on c. Then, (2) follows by induction on α_p. The case $\alpha_p = p_S(t_1, \ldots, t_n)$ does not arise because we assume that there are no subjective predicates. Similarly, the cases $\alpha_p = \Box\beta$ and $\alpha_p = \beta_1 \,\mathrm{U}\, \beta_2$ do not arise because α_p does not contain future operators. We show some other representative cases below.

Case. $\alpha_p = p_O(t_1, \ldots, t_n)$. Then, $(\alpha_p)^\tau = p_O(t_1, \ldots, t_n, \tau)$, which is T-protected because $\tau \in T$ is given.

Case. $\alpha_p = \neg\alpha'_p$. Then, $(\alpha_p)^\tau = \overline{(\alpha'_p)^\tau}$. By the i.h., $(\alpha'_p)^\tau$ is T-protected. Hence, by Lemma C.9, $\overline{(\alpha'_p)^\tau}$ is also T-protected.

Case. $\alpha_p = \forall \vec{x}.(c \supset \beta_p)$. Then, $(\alpha_p)^\tau = \forall \vec{x}.((c)^\tau \supset (\beta_p)^\tau)$. By statement (1) of the theorem, $(c)^\tau$ is T-protected, and by the i.h., $(\beta_p)^\tau$ is T-protected. Hence, $(\alpha_p)^\tau$ is T-protected by clause (6) of Defn C.6.

Case. $\alpha_p = \downarrow x.\beta_p$. Then, $(\alpha_p)^\tau = (\beta_p[\tau/x])^\tau$. By the i.h. on the smaller formula $\beta_p[\tau/x]$, we get that $(\beta_p[\tau/x])^\tau$ is T-protected.

Case. $\alpha_p = \beta_1 \,\mathrm{S}\, \beta_2$. Then, $(\alpha_p)^\tau = \exists \tau'.(\mathtt{in}(\tau', 0, \tau) \wedge (\beta_2)^{\tau'} \wedge (\forall \tau''.((\mathtt{in}(\tau'', \tau', \tau) \wedge \tau' \neq \tau'') \supset (\beta_1)^{\tau''})))$. First, by the i.h., $(\beta_1)^{\tau''}$ is $(T \cup \{\tau''\})$-protected. Consequently, by clause (7) of Defn C.6, $(\forall \tau''.((\mathtt{in}(\tau'', \tau', \tau) \wedge \tau' \neq \tau'') \supset (\beta_1)^{\tau''}))$ is T-protected. Hence, it is also $(T \cup \{\tau'\})$-protected. Call this fact (A). Next, by the i.h., $(\beta_2)^{\tau'}$ is $(T \cup \{\tau'\})$-protected. Combining this and fact (A), we have that $(\beta_2)^{\tau'} \wedge (\forall \tau''.((\mathtt{in}(\tau'', \tau', \tau) \wedge \tau' \neq \tau'') \supset (\beta_1)^{\tau''}))$ is $(T \cup \{\tau'\})$-protected. By clause (9) of Defn C.6, $(\alpha_p)^\tau$ is T-protected, as required.

Case. $\alpha_p = \boxminus\beta_p$. Then, $(\alpha_p)^\tau = \forall \tau'.(\mathtt{in}(\tau', \tau, \infty) \supset (\beta_p)^{\tau'})$. By the i.h., $(\beta_p)^{\tau'}$ is $(T \cup \{\tau'\})$-protected. Hence, by clause (7) of Defn C.6, $(\alpha_p)^\tau$ is T-protected. □

Lemma C.11 (Reduction of past formulas). *Let α_p be a temporal logic formula without future operators, and suppose that τ is a ground time point such that $\vdash (\alpha_p)^\tau$. Let $\mathcal{L}$ be τ_0-complete and $\tau_0 \geq \tau$. Then, either (1) $\mathsf{reduce}(\mathcal{L}, (\alpha_p)^\tau) \rightarrow^* \top$ and $\mathcal{L} \models (\alpha_p)^\tau$, or (2) $\mathsf{reduce}(\mathcal{L}, (\alpha_p)^\tau) \rightarrow^* \bot$ and $\mathcal{L} \models \overline{(\alpha_p)^\tau}$.*

Proof. By Lemma C.10(2), $(\alpha_p)^\tau$ is $\{\tau\}$-protected. Because $\tau \leq \tau_0$ and $\vdash (\alpha_p)^\tau$, by Lemma C.8, $\mathsf{reduce}(\mathcal{L}, (\alpha_p)^\tau) \to^* \psi$, where $\psi = \top$ or $\psi = \bot$ and $\mathcal{L} \models (\alpha_p)^\tau$ iff $\mathcal{L} \models \psi$. Call the latter fact (A). We consider two cases:

Case. $\psi = \top$. In this case, fact (A) means that $\mathcal{L} \models (\alpha_p)^\tau$ iff $\mathcal{L} \models \top$, which implies that $\mathcal{L} \models (\alpha_p)^\tau$. So (1) holds.

Case. $\psi = \bot$. In this case, fact (A) yields that $\mathcal{L} \not\models (\alpha_p)^\tau$. Since $(\alpha_p)^\tau$ is $\{\tau\}$-protected (already proved) and $\tau \leq \tau_0$, Lemma C.7(2) yields $\mathcal{L} \models \overline{(\alpha_p)^\tau}$. So (2) holds. □

Theorem C.12 (Enforcement of safety properties; Theorem 5.4)**.** *Suppose* $\mathbf{G}\,\alpha_p$ *is a safety property,* $\vdash \mathbf{G}\,\alpha_p$*,* $\mathcal{L}$ *is* τ_0*-complete, and for all* τ*,* $(\rho_{\mathcal{L}}(\mathtt{in}(\tau, 0, \infty)) = \mathtt{tt}) \Rightarrow \tau \leq \tau_0$*. Then,* $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\,\alpha_p) \to^* \bot$ *iff there is a* τ *such that* $\mathcal{L} \models \mathtt{in}(\tau, 0, \tau_0)$ *and* $\mathcal{L} \models \overline{(\alpha_p)^\tau}$*.*

Proof. We have $\mathbf{G}\,\alpha_p = \forall\tau.(\mathtt{in}(\tau, 0, \infty) \supset (\alpha_p)^\tau)$. Let $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\,\alpha_p) = \psi$. Then,

$$\begin{aligned}
\psi = \mathsf{reduce}(\mathcal{L}, \forall\tau.(\mathtt{in}(\tau,0,\infty) \supset (\alpha_p)^\tau)) \;=\; & \text{let} \\
& \quad \{\sigma_1, \ldots, \sigma_n\} \leftarrow \widehat{\mathtt{sat}}(\mathcal{L}, \mathtt{in}(\tau, 0, \infty)) \\
& \quad \{\tau_i \leftarrow \sigma_i(\tau)\}_{i=1}^n \\
& \quad S \leftarrow \{\tau_1, \ldots, \tau_n\} \\
& \quad \{\psi_i \leftarrow \mathsf{reduce}(\mathcal{L}, (\alpha_p)^{\tau_i})\}_{i=1}^n \\
& \quad \psi' \leftarrow \forall\tau.((\mathtt{in}(\tau, 0, \infty) \wedge \tau \notin S) \supset (\alpha_p)^\tau) \\
& \text{return} \\
& \quad \psi_1 \wedge \ldots \wedge \psi_n \wedge \psi'
\end{aligned}$$

By inversion on $\vdash \mathbf{G}\,\alpha_p$, we obtain a χ_O such that $\vdash \mathtt{in}(\tau, 0, \infty) : \chi_O$ and $\chi_O \vdash (\alpha_p)^\tau$. The first of these forces $\chi_O = \{\tau\}$, so from the second one we have that $\tau \vdash (\alpha_p)^\tau$. Using Lemma B.8(2), we get $\vdash (\alpha_p)^{\tau_i}$. Call this fact (A). Next, observe that by Theorem B.3, for each τ_i, $\mathcal{L} \models \mathtt{in}(\tau_i, 0, \infty)$, i.e., $\rho_{\mathcal{L}}(\mathtt{in}(\tau_i, 0, \infty)) = \mathtt{tt}$. This forces $\tau_i \leq \tau_0$ from the assumptions of the theorem we are trying to prove. Call this fact (B). We now prove the two directions of the conclusion of the theorem.

Direction "if". Suppose there is a τ with $\mathcal{L} \models \mathtt{in}(\tau, 0, \tau_0)$ and $\mathcal{L} \models \overline{(\alpha_p)^\tau}$. We prove that $\psi \to^* \bot$. By Theorem B.3 applied to $\mathcal{L} \models \mathtt{in}(\tau, 0, \tau_0)$, $\tau = \tau_i$ for some i. Hence by Lemma C.11, using facts (A) and (B) and $\mathcal{L} \models \overline{(\alpha_p)^\tau}$, we have that $\mathsf{reduce}(\mathcal{L}, (\alpha_p)^{\tau_i}) \to^* \bot$, i.e., $\psi_i \to^* \bot$. Clearly, $\psi = (\ldots \wedge \psi_i \wedge \ldots) \to^* \bot$, as required.

Direction "only if". Suppose that $\mathsf{reduce}(\mathcal{L}, \mathbf{G}\,\alpha_p) \to^* \bot$, i.e., $\psi \to^* \bot$. We show that there is a τ such that $\mathtt{in}(\tau, 0, \tau_0)$ and $\mathcal{L} \models \overline{(\alpha_p)^\tau}$. By definition of $\to$, we obtain that either for some i, $\psi_i \to^* \bot$ or $\psi' \to^* \bot$. The latter is impossible because ψ' has a top-level $\forall$, which can only be rewritten to $\top$. Hence, there is an i such that $\psi_i \to^* \bot$, i.e., $\mathsf{reduce}(\mathcal{L}, (\alpha_p)^{\tau_i}) \to^* \bot$. Choose $\tau = \tau_i$. By Lemma C.11, using facts (A) and (B) and $\mathsf{reduce}(\mathcal{L}, (\alpha_p)^{\tau_i}) \to^* \bot$, we obtain that $\mathcal{L} \models \overline{(\alpha_p)^{\tau_i}}$. The remaining requirement, $\mathcal{L} \models \mathtt{in}(\tau_i, 0, \tau_0)$ follows from fact (B). □

Theorem C.13 (Enforcement of co-safety properties; Theorem 5.5)**.** *Suppose* $\mathbf{F}\,\alpha_p$ *is a co-safety property,* $\vdash \mathbf{F}\,\alpha_p$*,* $\mathcal{L}$ *is* τ_0*-complete, and for all* τ*,* $(\rho_{\mathcal{L}}(\mathtt{in}(\tau, 0, \infty)) = \mathtt{tt}) \Rightarrow \tau \leq \tau_0$*. Then,* $\mathsf{reduce}(\mathcal{L}, \mathbf{F}\,\alpha_p) \to^* \top$ *if and only if there is a* τ *such that* $\mathcal{L} \models \mathtt{in}(\tau, 0, \tau_0)$ *and* $\mathcal{L} \models (\alpha_p)^\tau$*.*

Proof. Similar to that of Theorem C.12. □

D HIPAA Case study

This appendix lists the number of subjective and objective atoms in each transmission-related clause in the HIPAA Privacy Rule. #S denotes the number of subjective atoms; #O' denotes the number of such subjective atoms that can be mechanized by a small amount of design effort; and #O denotes the number of objective atoms. The table is sorted by the last column (#O' + #O) / (#S + #O).

Clause No.	#S	#O'	#O	(#O' + #O) / (#S + #O)
164.502(e)(1)(ii)(B)	0	0	5	1.00
164.502(a)(1)(i)	1	1	3	1.00
164.502(a)(1)(iv)	37	37	4	1.00
164.502(d)(1)	2	2	2	1.00
164.502(e)(1)(i)	1	1	2	1.00
164.508(a)(2)	37	37	4	1.00
164.508(a)(3)(i)	38	38	4	1.00
164.508(a)(3)(i)(A)	2	2	3	1.00
164.510(a)(1)(ii)	2	2	3	1.00
164.510(a)(2)	2	2	2	1.00
164.512(c)(2)	1	1	0	1.00
164.512(e)(1)(i)	3	3	4	1.00
164.512(e)(1)(ii)	9	9	4	1.00
164.512(e)(1)(vi)	4	4	2	1.00
164.512(f)(2)	10	10	3	1.00
164.512(f)(3)(i)	6	6	4	1.00
164.514(e)(1)	25	25	1	1.00
164.512(j)(3)	11	10	1	0.92
164.524(b)(2)(i)	54	43	41	0.88
164.524(b)(2)(ii)	53	42	42	0.88
164.512(g)(1)	4	3	4	0.88
164.510(b)(1)(i)	2	1	5	0.86
164.502(e)(1)(ii)(C)	3	2	3	0.83
164.506(c)(5)	8	6	4	0.83
164.512(b)(1)(v)	5	3	7	0.83
164.512(k)(1)(iii)	3	2	3	0.83
164.514(f)(1)	3	2	3	0.83
164.502(g)(3)(ii)(A)	2	1	4	0.83
164.502(g)(3)(ii)(B)	2	1	4	0.83
164.502(j)(2)	2	1	4	0.83
164.512(b)(1)(ii)	3	2	3	0.83
164.512(f)(5)	4	3	2	0.83
164.512(k)(1)(i)	2	1	4	0.83
164.512(k)(1)(iv)	2	1	4	0.83
164.512(k)(6)(i)	3	2	3	0.83
164.512(k)(6)(ii)	7	5	4	0.82
164.512(i)(1)	20	15	6	0.81

164.506(c)(3)	2	1	3	0.80
164.512(b)(1)(iii)	3	2	2	0.80
164.512(h)	2	1	3	0.80
164.512(k)(1)(ii)	4	3	1	0.80
164.512(g)(2)	4	2	5	0.78
164.512(d)(1)	6	4	3	0.78
164.502(a)(2)(ii)	2	1	2	0.75
164.506(c)(2)	2	1	2	0.75
164.510(b)(1)(ii)	4	2	4	0.75
164.512(b)(1)(iv)	3	2	1	0.75
164.510(b)(2)	5	3	3	0.75
164.512(f)(1)(i)	17	10	10	0.74
164.506(c)(4)	6	3	4	0.70
164.502(e)(1)(ii)(A)	1	0	2	0.67
164.506(b)(1)	1	0	2	0.67
164.506(c)(1)	4	2	2	0.67
164.512(f)(6)(i)	4	2	2	0.67
164.502(b)(1)	2	1	1	0.67
164.502(j)(1)	5	1	7	0.67
164.512(a)(1)	2	1	1	0.67
164.512(f)(1)(ii)	7	4	2	0.67
164.512(f)(4)	3	1	3	0.67
164.512(j)(1)(ii)(A)	18	11	3	0.67
164.512(l)	2	1	1	0.67
164.512(k)(4)	4	1	3	0.57
164.512(k)(3)	5	2	2	0.57
164.512(b)(1)(i)	6	1	5	0.55
164.502(b)(2)(i)	1	0	1	0.50
164.508(a)(2)(i)(B)	1	0	1	0.50
164.508(a)(2)(i)(C)	1	0	1	0.50
164.508(a)(3)(i)(B)	1	0	1	0.50
164.510(a)(3)(ii)	3	1	1	0.50
164.512(j)(1)(ii)(B)	4	1	2	0.50
164.512(k)(5)(i)	8	2	3	0.45
164.512(k)(2)	4	1	1	0.40
164.510(b)(4)	12	3	2	0.36
164.512(c)(1)	10	1	4	0.36
164.512(f)(3)(ii)	9	1	3	0.33
164.512(j)(1)(i)	5	1	1	0.33
164.514(g)	9	1	2	0.27
164.510(b)(3)	4	1	0	0.25
164.502(a)(1)(iii)	1	0	0	0.00
164.510(a)(3)(i)	4	0	0	0.00
164.512(c)(2)(i)	1	0	0	0.00
164.512(f)(6)(ii)	1	0	0	0.00

164.512(j)(2)(i)	1	0	0	0.00
164.512(j)(2)(ii)	1	0	0	0.00
Total	578	402	303	0.80
Clause No.	#S	#S’	#O	(#S’ + #O) / (#S + #O)